Cloudera Flow Management 4.11.0

Cloudera Flow Management Release Notes

Date published: 2019-06-26
Date modified: 2025-10-08

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

What's new in Cloudera Flow Management 4.11.0

Explore the latest features and improvements in Cloudera Flow Management and see how they help modernize your data pipeline development and operations.

Cloudera Flow Management 4.11.0 is built on Apache NiFi 2.4.0 and includes fixes addressing multiple Common Vulnerabilities and Exposures (CVEs) to enhance stability, reliability, and compliance for enterprise deployments. The following sections summarize the most important updates in this release.

New NiFi components

New processors:

The following new Box processors extend integration with the Box platform.

- CreateBoxFileMetadataInstance
- CreateBoxMetadataTemplate
- DeleteBoxFileMetadataInstance
- ExtractStructuredBoxFileMetadata
- FetchBoxFileMetadataInstance
- ListBoxFileMetadataInstances
- ListBoxFileMetadataTemplates
- UpdateBoxFileMetadataInstance

New controller services:

- ClouderaHiveConnectionPoolLookup

  You can dynamically select connections to Hive.
- JWTBearerOAuth2AccessTokenProvider

  You can support the JWT Bearer authentication approach for OAuth2.

New flow analysis rule:

- RequireServerSSLContextService

  You can enforce secure communication by ensuring that no server (for example, ListenHTTP or ListenTCP) can be created without an SSL context.

New flow registry client:

- BitbucketFlowRegistryClient

  You can enable BitBucket as a backend for NiFi Registry integration.

For a full list of supported NiFi components, see the Support Matrix

Other improvements

AWS MSK (Kafka) IAM authentication

It adds support for integrating with Kafka on AWS using IAM.

Astral uv integration

It enables dynamic downloading of Python dependencies. Similar to pipx but significantly faster, it provides command-line tools for managing Python environments and libraries.

New Migration Tool version available

Cloudera Flow Management Migration Tool 5.0.0. supports migrations to Cloudera Flow Management 4.11.0. For upgrade and migration options, see below. For details on the tool, see the Migration Tool documentation.

Upgrade and migration options

You can upgrade from Cloudera Flow Management 4.10.0 to 4.11.0 using Cloudera Manager. For step-by-step instructions, see Upgrading from Cloudera Flow Management 4.10.0 to 4.11.0.

In-place upgrades from Cloudera Flow Management 2.1.7 and lower versions to Cloudera Flow Management 4.11.0 are not supported. Flows created in NiFi 1 must be migrated to NiFi 2, and then imported into a Cloudera Flow Management 4.11.0 instance running NiFi 2. For more information on this migration process and the use of the Cloudera Flow Management Migration Tool, see the Migration documentation.

Important: To transition from Cloudera Flow Management versions powered by NiFi 1 to versions powered by NiFi 2 (4.10.0 and higher) using the Migration Tool, you must be using Cloudera Flow Management 2.1.7 SP2. For detailed version compatibility information for Apache NiFi and Cloudera Flow Management, see the Component versions page.

Installation

To perfrom a fresh installation of Cloudera Flow Management 4.11.0, follow the steps in the Cloudera Flow Management installation workflow.

Support matrix

Review the support matrix before you start installing Cloudera Flow Management.

Component versions

Review the Cloudera Flow Management component versions for compatibility with other applications.

Note: NiFi is compatible with the version of NiFi Registry bundled with your Cloudera Flow Management release as well as any later version.

Cloudera Flow Management 4.11.0

- Apache NiFi 2.4.0.4.11.0.0
- Apache NiFi Registry 2.4.0.4.11.0.0

Cloudera Flow Management 4.10.0

- Apache NiFi 2.3.0.4.10.0.0
- Apache NiFi Registry 2.3.0.4.10.0.0

Cloudera Flow Management 4.0.0 [Technical Preview]

- Apache NiFi 2.0.0.4.0.0.0
- Apache NiFi Registry 2.0.0.4.0.0.0

Cloudera Flow Management 2.1.7.2000 (SP2)

- Apache NiFi 1.28.1.2.1.7.2000
- Apache NiFi Registry 1.28.1.2.1.7.2000

Cloudera Flow Management 2.1.7.1000 (SP1)

- Apache NiFi 1.26.0.2.1.7.1000
- Apache NiFi Registry 1.26.0.2.1.7.1000

Cloudera Flow Management 2.1.7

- Apache NiFi 1.26.0.2.1.7.0
- Apache NiFi Registry 1.26.0.2.1.7.0

Cloudera Flow Management 2.1.6.1000 (SP1)

- Apache NiFi 1.23.1.2.1.6.1000
- Apache NiFi Registry 1.23.1.2.1.6.1000

Cloudera Flow Management 2.1.6

- Apache NiFi 1.23.1.2.1.6.0
- Apache NiFi Registry 1.23.1.2.1.6.0

Cloudera Flow Management 2.1.5.1000 (SP1)

- Apache NiFi 1.18.0.2.1.5.1000
- Apache NiFi Registry 1.18.0.2.1.5.1000

Cloudera Flow Management 2.1.5

- Apache NiFi 1.18.0.2.1.5.0
- Apache NiFi Registry 1.18.0.2.1.5.0

Cloudera Flow Management 2.1.4.1000 (SP1)

- Apache NiFi 1.16.0.2.1.4.1000
- Apache NiFi Registry 1.16.0.2.1.4.1000

Cloudera Flow Management 2.1.4

- Apache NiFi 1.16.0.2.1.4.0
- Apache NiFi Registry 1.16.0.2.1.4.0

Cloudera Flow Management 2.1.3

- Apache NiFi 1.15.2.2.1.3.0
- Apache NiFi Registry 1.15.2.2.1.3.0

Note: Apache NiFi and Apache NiFi Registry versions are unified in the 1.15.x release.

Cloudera Flow Management 2.1.2

- Apache NiFi 1.13.2.2.1.2.0
- Apache NiFi Registry 0.8.0.2.1.2.0

Cloudera Flow Management 2.1.1

- Apache NiFi 1.13.2.2.1.1.0
- Apache NiFi Registry 0.8.0.2.1.1.0

Cloudera Flow Management 2.0.4

- Apache NiFi 1.11.4
- Apache NiFi Registry 0.6.0

Cloudera Flow Management 2.0.1

- Apache NiFi 1.11.4
- Apache NiFi Registry 0.6.0

System requirements

Review the system requirements before getting started with installing Cloudera Flow Management 4.11.0.

Supported Cloudera platform versions

Cloudera Flow Management 4.11.0 supports the following versions of Cloudera Base on premises:

- 7.3.1 and all service packs

Supported JAVA Development Kits (JDK)

Cloudera Flow Management requires a minimum of JDK 21 for proper functionality. Ensure your environment meets this requirement before installation.

Other system requirements

ZooKeeper

You need to install the ZooKeeper service included with your Cloudera Base on premises cluster.

Python

- Minimum requirement: Python 3.11
- Recommended version: Python 3.12

Number of cores

- Minimum: Four cores per NiFi node are required for Cloudera support.
- Recommended: Eight cores per NiFi node, which typically provides an optimal starting point for most common use cases.

Supported operating systems

Review the list of operating systems supported in Cloudera Flow Management 4.11.0.

Operating system	Versions
RHEL	• 8.8 • 8.10 • 9.1 • 9.2 • 9.4
SLES	• 15 SP5
Oracle	• 8.8
Ubuntu	• 20.04 • 22.04
Windows	• 10 • Server 2016 • Server 2019 • Server 2022 • Server 2025

Note:

NiFi on Windows is only supported in standalone mode, not managed by Cloudera Manager or as part of a Cloudera cluster, and as a single instance installation. Clustering NiFi on Windows is not supported.

NiFi Registry is not supported on Windows.

Supported NiFi Registry databases

Review the list of databases supported by NiFi Registry in Cloudera Flow Management 4.11.0.

- H2
- PostgreSQL 10.x
- PostgreSQL 11.x
- PostgreSQL 12.x
- PostgreSQL 13.x
- PostgreSQL 14.x
- MySQL 8.x

Supported NiFi processors

Review the list of Apache NiFi processors supported in Cloudera Flow Management 4.11.0.

Cloudera Flow Management is based on Apache NiFi and includes a set of processors, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported processors and avoid deploying unsupported ones in production environments.

Additional processors are developed and tested by the community but are not officially supported by Cloudera. Processors may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

AttributesToCSV	GetAwsTranslateJobStatus
AttributesToJSON	GetAzureEventHub
CalculateParquetOffsets	GetAzureQueueStorage_v12
CalculateParquetRowGroupOffsets	GetBoxFileCollaborators
CalculateRecordStats	GetBoxGroupMembers
CaptureChangeDebeziumDB2	GetCouchbaseKey1
CaptureChangeDebeziumMongoDB	GetElasticsearch
CaptureChangeDebeziumMySQL	GetFile
CaptureChangeDebeziumOracle	GetFTP
CaptureChangeDebeziumPostgreSQL	GetGcpVisionAnnotateFilesOperationStatus
CaptureChangeDebeziumSQLServer	GetGcpVisionAnnotateImagesOperationStatus
CaptureChangeMySQL	GetHBase
ChunkDocument	GetHDFS
CompressContent1, 2	GetHDFSFileInfo
ConnectWebSocket	GetHDFSSequenceFile
ConsumeAMQP	GetHubSpot
ConsumeAzureEventHub	GetJiraIssue

ConsumeBoxEnterpriseEvents	GetMongoRecord
ConsumeBoxEvents	GetS3ObjectTags
ConsumeElasticsearch	GetSFTP
ConsumeGCPubSub	GetShopify
ConsumeGCPubSubLite	GetSNMP
ConsumeJMS	GetSnowflakeIngestStatus
ConsumeKafka_2_6	GetSolr
ConsumeKafka2CDP	GetSplunk
ConsumeKafka2RecordCDP	GetSQS
ConsumeKafkaRecord_2_6	GetWorkdayReport
ConsumeKinesisStream	GetZendesk
ConsumeMQTT1	HandleHttpRequest
ConsumePLC	HandleHttpResponse
ConsumeSlack	IdentifyMimeType
ConsumeTwitter	InvokeAWSGatewayApi
ConsumeWindowsEventLog	InvokeGRPC
ControlRate	InvokeHTTP
ConvertAvroToJSON	InvokeScriptedProcessor
ConvertAvroToParquet	JoinEnrichment
ConvertCharacterSet	JoltTransformJSON
ConvertJSONToSQL	JoltTransformRecord
ConvertProtobuf	JSLTTransformJSON
ConvertRecord	JsonQueryElasticsearch
CopyAzureBlobStorage_v12	ListAzureBlobStorage_v12
CountText	ListAzureDataLakeStorage
CreateBoxFileMetadataInstance	ListBoxFile
CreateBoxMetadataTemplate	ListBoxFileInfo
CreateHadoopSequenceFile	ListBoxFileMetadataInstances
CryptographicHashContent	ListBoxFileMetadataTemplates
DecryptContent	ListCDPObjectStore
DecryptContentAge	ListDatabaseTables
DecryptContentCompatibility	ListDropbox
DecryptContentPGP	ListenBeats
DeduplicateRecord	ListenFTP
DeleteAzureBlobStorage_v12	ListenGRPC
DeleteAzureDataLakeStorage	ListenHTTP
DeleteBoxFileMetadataInstance	ListenNetFlow
DeleteByQueryElasticsearch	ListenOTLP
DeleteCDPObjectStore	ListenRELP
DeleteDynamoDB	ListenSlack

DeleteGCSObject	ListenSyslog
DeleteGridFS	ListenTCP
DeleteHBaseCells	ListenTCPRecord
DeleteHBaseRow	ListenTrapSNMP
DeleteHDFS	ListenUDP
DeleteS3Object	ListenUDPRecord
DeleteSQS	ListenWebSocket
DetectDuplicate	ListFile
DistributeLoad	ListFTP
DuplicateFlowFile	ListGCSBucket
EncodeContent	ListGoogleDrive
EncryptContentAge	ListHDFS
EncryptContentPGP	ListS3
EnforceOrder	ListSFTP
EvaluateJsonPath	ListSmb
EvaluateXPath	LogAttribute
EvaluateXQuery	LogMessage
ExecuteGroovyScript	LookupAttribute
ExecuteProcess	LookupRecord
ExecuteScript	MergeContent
ExecuteSQL	MergeRecord1
ExecuteSQLRecord	ModifyCompression
ExecuteStateless1, 2	MonitorActivity
ExecuteStreamCommand	MoveAzureDataLakeStorage
ExtractAvroMetadata	MoveHDFS
ExtractGrok	Notify
ExtractHL7Attributes	PackageFlowFile
ExtractImageMetadata	PaginatedJsonQueryElasticsearch
ExtractRecordSchema	ParseCEF1
ExtractStructuredBoxFileMetadata	ParseDocument
ExtractText	ParseEvtx
FetchAzureBlobStorage_v12	ParseSyslog
FetchAzureDataLakeStorage	PartitionRecord
FetchBoxFile	PromptAzureOpenAI
FetchBoxFileInfo	PromptChatGPT
FetchBoxFileMetadataInstance	PromptClaude
FetchBoxFileRepresentation	PromptOpenAI
FetchCDPObjectStore	PublishAMQP
FetchDistributedMapCache	PublishGCPubSub1
FetchDropbox	PublishGCPubSubLite1

FetchFile	PublishJMS1
FetchFTP	PublishKafka_2_6
FetchGCSObject	PublishKafka2CDP
FetchGoogleDrive	PublishKafka2RecordCDP
FetchGridFS	PublishKafkaRecord_2_6
FetchHBaseRow	PublishMQTT
FetchHDFS	PublishSlack
FetchParquet	PutAccumuloRecord1
FetchPLC	PutAzureBlobStorage_v12
FetchS3Object	PutAzureCosmosDBRecord
FetchSFTP	PutAzureDataLakeStorage1
FetchSmb	PutAzureEventHub
FilterAttribute	PutAzureQueueStorage_v12
FlattenJson	PutBigQuery
ForkEnrichment	PutBoxFile
ForkRecord	PutCassandraQL1
GenerateFlowFile	PutCassandraRecord1
GenerateRecord	PutCDPObjectStore
GenerateTableFetch	PutChroma
GeoEnrichIP	PutClouderaHiveQL
GeoEnrichIPRecord	PutClouderaHiveStreaming
GeohashRecord	PutClouderaORC
GetAsanaObject	PutCloudWatchMetric
GetAwsPollyJobStatus	PutCouchbaseKey
GetAwsTextractJobStatus	PutDatabaseRecord1
GetAwsTranscribeJobStatus	PutDistributedMapCache
	PutDropbox

Footnotes

- 1 – indicates a memory intensive processor
- 2 – indicates a CPU intensive processor

Related Information

Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi controller services

Review the list of Apache NiFi controller services supported in Cloudera Flow Management 4.11.0.

Cloudera Flow Management is based on Apache NiFi and includes a set of controller services, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported controller services and avoid deploying unsupported ones in production environments.

Additional controller services are developed and tested by the community but are not officially supported by Cloudera. Controller services may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

AccumuloService	IPFIXReader
ActiveMQJMSConnectionFactoryProvider	IPLookupService
ADLSCredentialsControllerService	JASN1Reader
ADLSCredentialsControllerServiceLookup	JiraRecordSink
ADLSIDBrokerCloudCredentialsProviderControllerService	JMSConnectionFactoryProvider
AmazonGlueSchemaRegistry	JndiJmsConnectionFactoryProvider
ApicurioSchemaRegistry	JsonConfigBasedBoxClientService
AvroReader	JsonPathReader
AvroRecordSetWriter	JsonRecordSetWriter
AvroSchemaRegistry	JsonTreeReader
AWSCredentialsProviderControllerService	JWTBearerOAuth2AccessTokenProvider
AWSIDBrokerCloudCredentialsProviderControllerService	KafkaRecordSink_2_6
AzureBlobIDBrokerCloudCredentialsProviderControllerService	KerberosKeytabUserService
AzureCosmosDBClientService	KerberosPasswordUserService
AzureEventHubRecordSink	KerberosTicketCacheUserService
AzureServiceBusJMSConnectionFactoryProvider	KuduLookupService
AzureStorageCredentialsControllerService_v12	LoggingRecordSink
AzureStorageCredentialsControllerServiceLookup_v12	MongoDBControllerService
CassandraDistributedMapCache	MongoDBLookupService
CassandraSessionProvider	ParquetReader
CdpCredentialsProviderControllerService	ParquetRecordSetWriter
CdpOauth2AccessTokenProviderControllerService	PEMEncodedSSLContextProvider
CEFReader	PhoenixThickConnectionPool
CiscoEmblemSyslogMessageReader	PhoenixThinConnectionPool
ClouderaAttributeSchemaReferenceReader	PostgreSQLConnectionPool
ClouderaAttributeSchemaReferenceWriter	PrometheusRecordSink
ClouderaEncodedSchemaReferenceReader	ProxyPLC4XConnectionPool
ClouderaEncodedSchemaReferenceWriter	RabbitMQJMSConnectionFactoryProvider
ClouderaHiveConnectionPool	ReaderLookup
ClouderaHiveConnectionPoolLookup	RecordSetWriterLookup
ClouderaSchemaRegistry	RecordSinkServiceLookup
CMLLookupService	RedisConnectionPoolService
ConfluentEncodedSchemaReferenceReader	RedisDistributedMapCacheClientService
ConfluentEncodedSchemaReferenceWriter	RedshiftConnectionPool

ConfluentSchemaRegistry	RESTCatalogService
CouchbaseClusterService	RestLookupService
CouchbaseKeyValueLookupService	ScriptedLookupService
CouchbaseMapCacheClient	ScriptedReader
CouchbaseRecordLookupService	ScriptedRecordSetWriter
CSVReader	ScriptedRecordSink
CSVRecordLookupService	SimpleDatabaseLookupService
CSVRecordSetWriter	SimpleKeyValueLookupService
DatabaseRecordLookupService	SimpleRedisDistributedMapCacheClientServic
DatabaseRecordSink	SimpleScriptedLookupService
DatabaseTableSchemaRegistry	SiteToSiteReportingRecordSink
DeveloperBoxClientService	SlackRecordSink
DBCPConnectionPool	SmbjClientProviderService
DBCPConnectionPoolLookup	SnowflakeComputingConnectionPool
DistributedMapCacheClientService	StandardAsanaClientProviderService
DistributedMapCacheLookupService	StandardAzureCredentialsControllerService
DistributedMapCacheServer	StandardDatabaseDialectService
DistributedSetCacheClientService	StandardDropboxCredentialService
DistributedSetCacheServer	StandardFileResourceService
EBCDICRecordReader [Technical Preview]	StandardHashiCorpVaultClientService
ElasticSearchClientServiceImpl	StandardHttpContextMap
ElasticSearchLookupService	StandardJsonSchemaRegistry [Technical Previe
ElasticSearchStringLookupService	StandardOauth2AccessTokenProvider
EmailRecordSink	StandardPGPPrivateKeyService
EmbeddedHazelcastCacheManager	StandardPGPPublicKeyService
ExcelReader	StandardPLC4XConnectionPool
ExternalHazelcastCacheManager	StandardPrivateKeyService
FreeFormTextRecordSetWriter	StandardProxyConfigurationService
GCPCredentialsControllerService	StandardRestrictedSSLContextService
GCSFileResourceService	StandardS3EncryptionService
GenericPLC4XConnectionPool	StandardSnowflakeIngestManagerProviderServ
GrokReader	StandardSSLContextService
HadoopCatalogService	StandardWebClientServiceProvider
HadoopDBCPConnectionPool	Syslog5424Reader
HazelcastMapCacheClient	SyslogReader
HBase_2_ClientMapCacheService	UDPEventRecordSink
HBase_2_ClientService	VolatileSchemaCache
HBase_2_RecordLookupService	WindowsEventLogReader
Hive3ConnectionPool	XMLReader
HiveCatalogService	XMLRecordSetWriter

ImpalaConnectionPool	YamlTreeReader
	ZendeskRecordSink

Related Information

Supported NiFi processors
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi reporting tasks

Review the list of Apache NiFi reporting tasks supported in Cloudera Flow Management 4.11.0.

Cloudera Flow Management is based on Apache NiFi and includes a set of reporting tasks, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported reporting tasks and avoid deploying unsupported ones in production environments.

- ControllerStatusReportingTask
- MonitorDiskUsage
- MonitorMemory
- PrometheusReportingTask
- QueryNiFiReportingTask
- ReportLineageToAtlas
- ScriptedReportingTask
- SiteToSiteBulletinReportingTask
- SiteToSiteMetricsReportingTask
- SiteToSiteProvenanceReportingTask
- SiteToSiteStatusReportingTask

Additional reporting tasks are developed and tested by the community but are not officially supported by Cloudera. Reporting tasks may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi parameter providers
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi parameter providers

Review the list of Apache NiFi parameter providers supported in Cloudera Flow Management 4.11.0.

Cloudera Flow Management is shipped with Apache NiFi and includes a set of parameter providers, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported parameter providers and avoid deploying unsupported ones in production environments.

- AwsSecretsManagerParameterProvider
- AzureKeyVaultSecretsParameterProvider
- CyberArkConjurParameterProvider
- DatabaseParameterProvider

- EnvironmentVariableParameterProvider
- FileParameterProvider
- GcpSecretManagerParameterProvider
- HashiCorpVaultParameterProvider
- OnePasswordParameterProvider
- PropertiesFileParameterProvider

Additional parameter providers are developed and tested by the community but are not officially supported by Cloudera. Parameter providers may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi flow analysis rules

Review the list of Apache NiFi processors supported in Cloudera Flow Management 4.11.0.

Flow Analysis Rules allow analyzing components or parts of a flow to help maintain optimal design. Rules can be set as Recommendations (informational only) or Policies (enforceable). Recommendation violations are logged but do not affect functionality, while Policy violations invalidate components until resolved.

- DisallowComponentType
- DisallowConsecutiveConnectionsWithRoundRobinLB
- DisallowDeadEnd
- DisallowDeprecatedProcessor
- DisallowExtractTextForFullContent
- RecommendRecordProcessor
- RequireHandleHttpResponseAfterHandleHttpRequest
- RequireMergeBeforePutIceberg
- RequireServerSSLContextService
- RestrictBackpressureSettings
- RestrictComponentNaming
- RestrictConcurrentTasksVsThreadPoolSizeInProcessors
- RestrictFlowFileExpiration
- RestrictProcessorConcurrency
- RestrictSchedulingForListProcessors
- RestrictThreadPoolSize
- RestrictYieldDurationForConsumeKafkaProcessors

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow registry clients

Supported NiFi flow registry clients

Review the list of Apache NiFi flow registry clients supported in Cloudera Flow Management 4.11.0.

NiFi Flow Registry clients are components in Apache NiFi that allow it to connect to external Flow Registries, services used to store and manage versioned dataflows.

- BitbucketFlowRegistryClient
- ClouderaDataFlowRegistryClient
- ClouderaFlowLibraryFlowRegistryClient
- GitHubFlowRegistryClient
- GitLabFlowRegistryClient
- NifiRegistryFlowRegistryClient

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow analysis rules

Supported Cloudera exclusive components

Review the list of Cloudera exclusive components supported in Cloudera Flow Management 4.11.0.

Cloudera Flow Management provides a set of NiFi components available only to Cloudera customers. These components provide additional functionality and are tailored to enhance the Cloudera NiFi experience. The list of these components is provided below.

Processors

- CaptureChangeDebeziumDB2
- CaptureChangeDebeziumMongoDB
- CaptureChangeDebeziumMySQL
- CaptureChangeDebeziumOracle
- CaptureChangeDebeziumPostgreSQL
- CaptureChangeDebeziumSQLServer
- ConsumeKafka2CDP
- ConsumeKafka2RecordCDP
- ConsumePLC
- ConvertProtobuf
- DeleteCDPObjectStore
- FetchCDPObjectStore
- FetchPLC
- GetJiraIssue
- InvokeGRPC
- ListCDPObjectStore
- ListenGRPC
- ListenNetFlow
- PromptAzureOpenAI
- PromptBedrock
- PromptClaude
- PromptOpenAI

- PublishKafka2CDP
- PublishKafka2RecordCDP
- PutCDPObjectStore
- PutClouderaHiveQL
- PutClouderaHiveStreaming
- PutClouderaORC
- PutIcebergCDC
- PutJiraIssue
- PutPLC
- SawmillTransformJSON
- SawmillTransformRecord
- SelectClouderaHiveQL
- TokenCount
- TriggerClouderaHiveMetaStoreEvent
- UpdateClouderaHiveTable
- UpdateDeltaLakeTable

Controller services

- ActiveMQJMSConnectionFactoryProvider
- ADLSIDBrokerCloudCredentialsProviderControllerService
- AWSIDBrokerCloudCredentialsProviderControllerService
- AzureBlobIDBrokerCloudCredentialsProviderControllerService
- AzureServiceBusJMSConnectionFactoryProvider
- CdpCredentialsProviderControllerService
- CdpOauth2AccessTokenProviderControllerService
- CiscoEmblemSyslogMessageReader
- ClouderaAttributeSchemaReferenceReader
- ClouderaAttributeSchemaReferenceWriter
- ClouderaEncodedSchemaReferenceReader
- ClouderaEncodedSchemaReferenceWriter
- ClouderaHiveConnectionPool
- ClouderaSchemaRegistry
- CMLLookupService
- EBCDICRecordReader
- GenericPLC4XConnectionPool
- ImpalaConnectionPool
- IPFIXReader
- JiraRecordSink
- PhoenixThickConnectionPool
- PhoenixThinConnectionPool
- PostgreSQLConnectionPool
- ProxyPLC4XConnectionPool
- RabbitMQJMSConnectionFactoryProvider
- RedshiftConnectionPool
- RESTCatalogService
- StandardPLC4XConnectionPool

Parameter providers

- CyberArkConjurParameterProvider

- PropertiesFileParameterProvider

Flow registry clients

- ClouderaDataFlowRegistryClient
- ClouderaFlowLibraryFlowRegistryClient

Download locations

You can download the Cloudera Flow Management software artifacts from the Cloudera Archive. There are different artifacts for different operating systems, standalone components, and Windows files.

Use the following tables to identify the Cloudera Flow Management repository location for your operating system and operational objectives.

Note:

You must have credentials to download Cloudera Flow Management files. Your download credential is not the same as the credential you use to access the Cloudera Support Portal.

You can get download credentials in the following ways:

- Contact your Cloudera sales representative.
- Check the Welcome email you have received for your Cloudera Flow Management account.
- File a non-technical case on the Cloudera Support Portal for the Cloudera Support team to assist you.

Table 1: RHEL 8

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/parcel/CFM-2.4.0.4.11.0.0-352.parcel
Parcel sha file	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/parcel/CFM-2.4.0.4.11.0.0-352.parcel.sha
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/parcel/NIFI-2.4.0.4.11.0.0-352.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/parcel/NIFIREGISTRY-2.4.0.4.11.0.0-352.jar

Table 2: RHEL 9

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat9/yum/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat9/yum/tars/parcel/CFM-2.4.0.4.11.0.0-352.parcel
Parcel sha file	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat9/yum/tars/parcel/CFM-2.4.0.4.11.0.0-352.parcel.sha
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat9/yum/tars/parcel/NIFI-2.4.0.4.11.0.0-352.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat9/yum/tars/parcel/NIFIREGISTRY-2.4.0.4.11.0.0-352.jar

Table 3: Ubuntu 20

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu20/apt/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu20/apt/tars/parcel/CFM-2.4.0.4.11.0.0-352-focal.parcel
Parcel SHA file	https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu20/apt/tars/parcel/CFM-2.4.0.4.11.0.0-352-focal.parcel.sha
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu20/apt/tars/parcel/NIFI-2.4.0.4.11.0.0-352.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu20/apt/tars/parcel/NIFIREGISTRY-2.4.0.4.11.0.0-352.jar

Table 4: Ubuntu 22

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu22/apt/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu22/apt/tars/parcel/CFM-2.4.0.4.11.0.0-352-focal.parcel
Parcel SHA file	https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu22/apt/tars/parcel/CFM-2.4.0.4.11.0.0-352-focal.parcel.sha
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu22/apt/tars/parcel/NIFI-2.4.0.4.11.0.0-352.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.11.0.0/ubuntu22/apt/tars/parcel/NIFIREGISTRY-2.4.0.4.11.0.0-352.jar

Table 5: SLES 15

File	Location
Manifest	
Parcel	
Parcel SHA file	
CSD	**NiFi:** **NiFi Registry:**

Table 6: Standalone components (OS agnostic)

File	Location
NiFi (.tar.gz)	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/cdf_extensions/nifi-2.4.0.4.11.0.0-352-bin.tar.gz
NiFi (.tar.gz.sha256)	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/cdf_extensions/nifi-2.4.0.4.11.0.0-352-bin.tar.gz.sha256
NiFi Registry (.tar.gz)	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/cdf_extensions/nifi-registry-2.34.0.4.11.0.0-352-bin.tar.gz
NiFi Registry (.zip)	https://archive.cloudera.com/p/cfm2/4.11.0.0/redhat8/yum/tars/cdf_extensions/nifi-registry-2.4.0.4.11.0.0-352-bin.zip

Table 7: Windows files

File	Location
NiFi MSI	https://archive.cloudera.com/p/cfm2/4.11.0.0/windows/nifi-4.11.0.0-352.msi
NiFi MSI SHA	https://archive.cloudera.com/p/cfm2/4.11.0.0/windows/nifi-4.11.0.0-352.msi.sha256

Unsupported features

The following features are developed and tested by the Cloudera community but are not officially supported by Cloudera. These features are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Cloudera best practices. Do not use these features in your production environments.

Behavioral changes

Learn about behavioral changes in Cloudera Flow Management 4.11.0.

NiFi 2 introduces a lot of significant changes and enhancements, including some breaking changes. It is important to familiarize yourself with the following points before migrating your existing flows.

Java 21

Java 21 is the minimum Java version required with NiFi 2, so make sure you have Java 21 installed on your NiFi nodes before upgrading.

Templates and XML flow definitions

The concept of templates in NiFi has been deprecated, and the XML templates are stored in memory in NiFi as well as in the persisted flow definition.

Additionally, flow.xml.gz no longer exists, only flow.json.gz can be used in NiFi clusters for defining flows in the canvas.

If you have templates, export those templates as JSON definitions or version the templates into a NiFi Registry instance. The best practice is to use a NiFi Registry in combination with NiFi when it comes to version control and share / reuse flow definitions.

Custom components / NARs

Although not certain, it is very likely that a custom NAR designed for NiFi 1 will not be successfully loaded into NiFi 2. If your NiFi setup includes custom components or NARs, it is a requirement to update your dependencies to align with NiFi 2. This entails making the necessary adjustments and rebuilding your NARs using Java 21.

Variables replaced by parameters

Variables and the variable registry have been removed from NiFi. Only Parameter contexts and parameters are available for use going forward. In future Cloudera Flow Management releases, tooling will be provided to help with the conversion of variables to parameters. In the meantime, this conversion should be done manually when migrating flows to NiFi 2. Any variables left will simply be ignored when loading the flow definition.

Event driven thread pool no longer exists

The event driven scheduling strategy was an option available on some processors. This was an experimental feature in NiFi and did not prove to bring any significant performance improvements. The event driven thread pool has been removed, leaving only the time driven thread pool available.

Any components previously configured using the event driven scheduling strategy should be switched to the time driven scheduling strategy.

Removed languages in scripted components

In NiFi 2.0, support for certain languages in scripted components has been removed. The affected languages are: ECMAScript, Lua, Ruby, and Python. It is recommended to switch to Groovy or to leverage the new Python API feature for developing processors.

Removed components and replacement options

The following list contains the list of the components that have been removed between clusters based on NiFi 1 and clusters based on a NiFi 2 version, along with the recommended alternatives where available.

- Processors
  - Base64EncodeContent => EncodeContent
  - CompareFuzzyHash => no replacement
  - ConsumeEWS => no replacement
  - ConsumeKafka_1_0 => ConsumeKafka_2_6
  - ConsumeKafka_2_0 => ConsumeKafka_2_6
  - ConsumeKafkaRecord_1_0 => ConsumeKafkaRecord_2_6
  - ConsumeKafkaRecord_2_0 => ConsumeKafkaRecord_2_6
  - ConvertAvroSchema => ConvertRecord
  - ConvertAvroToORC => no replacement
  - ConvertCSVToAvro => ConvertRecord
  - ConvertExcelToCSVProcessor => ConvertRecord with ExcelReader
  - ConvertJSONToAvro => ConvertRecord
  - CryptographicHashAttribute => UpdateAttribute
  - DeleteAzureBlobStorage => DeleteAzureBlobStorage_v12
  - DeleteRethinkDB => no replacement
  - EncryptContent => EncryptContentAge or EncryptContentPGP
  - ExecuteInfluxDBQuery => use Influx Data NARs for NiFi
  - ExtractCCDAAttributes => no replacement
  - FetchAzureBlobStorage => FetchAzureBlobStorage_v12
  - FetchElasticsearchHttp => GetElasticsearch
  - FuzzyHashContent => no replacement
  - GetAzureQueueStorage => GetAzureQueueStorage_v12
  - GetHTMLElement => no replacement
  - GetHTTP => InvokeHTTP
  - GetIgniteCache => no replacement
  - GetJMSQueue => ConsumeJMS
  - GetJMSTopic => ConsumeJMS
  - GetRethinkDB => no replacement
  - GetTCP => no replacement
  - GetTwitter => ConsumeTwitter
  - HashAttribute => CryptographicHashAttribute
  - HashContent => CryptographicHashContent
  - InferAvroSchema => ExtractRecordSchema
  - ListAzureBlobStorage => ListAzureBlobStorage_v12
  - ModifyHTMLElement => no replacement
  - PostHTTP => InvokeHTTP
  - PostSlack => PublishSlack
  - PublishKafka_1_0 => PublishKafka_2_6
  - PublishKafka_2_0 => PublishKafka_2_6
  - PublishKafkaRecord_1_0 => PublishKafkaRecord_2_6
  - PublishKafkaRecord_2_0 => PublishKafkaRecord_2_6
  - PutAzureBlobStorage => PutAzureBlobStorage_v12
  - PutAzureQueueStorage => PutAzureQueueStorage_v12
  - PutBigQueryBatch => PutBigQuery
  - PutBigQueryStreaming => PutBigQuery
  - PutElasticsearchHttp => PutElasticsearchJson
  - PutElasticsearchHttpRecord => PutElasticsearchRecord
  - PutHiveQL => PutClouderaHiveQL
  - PutHiveStreaming => PutClouderaHiveStreaming

    - PutHTMLElement => no replacement
    - PutIgniteCache => no replacement
    - PutInfluxDB => use Influx Data NARs for NiFi
    - PutJMS => PublishJMS
    - PutRethinkDB => no replacement
    - PutRiemann => no replacement
    - PutSlack => PublishSlack
    - QueryElasticsearchHttp => PaginatedJsonQueryElasticsearch
    - ScrollElasticsearchHttp => SearchElasticsearch
    - SelectHiveQL => SelectClouderaHiveQL
    - SpringContextProcessor => no replacement
    - StoreInKiteDataset => no replacement
    - UpdateHiveTable => UpdateClouderaHiveTable
- Controller services
    - ActionHandlerLookup => no replacement
    - AlertHandler => no replacement
    - AzureStorageCredentialsControllerService => AzureStorageCredentialsControllerService_v12
    - AzureStorageCredentialsControllerServiceLookup => AzureStorageCredentialsControllerServiceLookup_v12
    - AzureStorageEmulatorCredentialsControllerService => no replacement
    - EasyRulesEngineProvider => no replacement
    - EasyRulesEngineService => no replacement
    - ExpressionHandler => no replacement
    - GraphiteMetricReporterService => no replacement
    - GremlinClientService => no replacement
    - HBase_1_1_2_ClientMapCacheService => HBase_2_ClientMapCacheService
    - HBase_1_1_2_ClientService => HBase_2_ClientService
    - HBase_1_1_2_ListLookupService => no replacement
    - HBase_1_1_2_RecordLookupService => HBase_2_RecordLookupService
    - HiveConnectionPool => ClouderaHiveConnectionPool
    - HortonworksSchemaRegistry => ClouderaSchemaRegistry
    - KafkaRecordSink_1_0 => KafkaRecordSink_2_6
    - KafkaRecordSink_2_0 => KafkaRecordSink_2_6
    - KeytabCredentialsService => KerberosKeytabUserService
    - LogHandler => no replacement
    - OAuth2TokenProviderImpl => StandardOauth2AccessTokenProvider
    - OpenCypherClientService => no replacement
    - RecordSinkHandler => no replacement
    - ScriptedActionHandler => no replacement
    - ScriptedRulesEngine => no replacement
- Reporting tasks
    - AmbariReportingTask => no replacement
    - MetricsEventReportingTask => no replacement
    - MetricsReportingTask => no replacement

- Components with new coordinates
  - InvokeGRPC => moved into nifi-cdf-grpc-nar
  - ListenGRPC => moved into nifi-cdf-grpc-nar
  - KerberosKeytabUserService => moved into nifi-kerberos-user-service-nar
  - KerberosPasswordUserService => moved into nifi-kerberos-user-service-nar
  - KerberosTicketCacheUserService => moved into nifi-kerberos-user-service-nar

  A migration tool is provided to help with handling these changes. For more information, see the Cloudera Flow Management Migration Tool documentation.
- Pulsar components

  All Pulsar components have been removed. You can download the NARs from a public Maven repository and deploy them as custom NARs.
  - nifi-pulsar-nar
  - nifi-pulsar-client-service-nar

Breaking changes in NiFi 2

Learn about the breaking changes Apache NiFi 2 will bring to Cloudera Flow Management.

Flow design level breaking changes

Moved components

In NiFi 2, some components have been relocated within the Apache NiFi repository, resulting in changes to the bundle coordinates for the associated NAR files.

Unfortunately, no pre-upgrade actions can fully prevent these breaking changes. You will need to update the flow.json.gz file with new coordinates. Cloudera's upcoming NiFi Migration Tooling is designed to automate as many changes as possible to help the upgrade process. Some changes will still require manual handling, so it is highly recommended to run the pre-upgrade check script to identify any potential issues or impacted components before proceeding with the upgrade.

JoltTransformJSON processor

For Previous coordinates

```
…
            "type": "org.apache.nifi.processors.standard.JoltT
ransformJSON",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-standard-nar",
              "version": "1.27.0"
            }
…
```

For New coordinates

```
…
            "type": "org.apache.nifi.processors.jolt.JoltTrans
formJSON",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-jolt-nar",
              "version": "2.0.0"
            }
```

```
…
```

JoltTransformRecord processor

For Previous coordinates

```
…
            "type": "org.apache.nifi.processors.jolt.record.Jo
ltTransformRecord",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-jolt-record-nar",
              "version": "1.27.0"
            }
…
```

For New coordinates

```
…
            "type": "org.apache.nifi.processors.jolt.JoltTrans
formRecord",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-jolt-nar",
              "version": "2.0.0"
            }
…
```

FileParameterProvider renamed to KubernetesSecretParameterProvider

For Previous coordinates

```
…
            "type": "org.apache.nifi.parameter.FileParameterPr
ovider",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-standard-nar",
              "version": "1.27.0"
            }
…
```

For New coordinates

```
…
            "type": "org.apache.nifi.parameter.KubernetesSecre
tParameterProvider",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-standard-nar",
              "version": "2.0.0"
            }
…
```

Removed key components

- Kafka processors:

  All Kafka processors in Apache NiFi have been removed and got replaced by new components using a controller service-based approach. This change is a significant breaking change, as it does not allow for a non-breaking upgrade. To ease this transition, Cloudera has preserved the Kafka 2.6 processors without altering the bundle coordinates. However, adjustments to the Kerberos configuration will still be necessary (see the details below). This approach provides time to transition to the new Kafka components while on NiFi 2.
- Hive components:

  All Hive-related components in Apache NiFi have been removed. Cloudera has introduced specific components downstream to align with the Hive version distributed as part of CDP. You will need to perform proper bundle coordinate updates in the flow.json.gz file to migrate to these new components. Both the old and new components are available in the latest NiFi 1.x releases, so you are advised to switch to the new components while still using NiFi 1.x.

Components with Kerberos configuration changes

Kerberos authentication in NiFi requires presenting a Kerberos credential which can be in one of the following forms:

- Principal + Keytab: The keytab, stored on disk, contains the client's secret key. This credential is used by the application to authenticate and obtain the Ticket Granting Ticket (TGT).
- Principal + Password: The client's secret key is derived from a password, so no keytab is stored on disk. Otherwise, this method functions similarly to the keytab-based approach.
- Principal + Ticket cache: The TGT must be acquired externally and stored in a ticket cache, which the application uses. The application itself is unaware of the keytab or password and is not responsible for handling TGT acquisition.

Historically, NiFi supported the following Kerberos configuration options:

- Kerberos Principal + Kerberos Keytab: Component-level properties supporting keytab-based credential type
- Kerberos Credentials Service: A property referencing the KerberosCredentialsService controller service interface with the following implementation: KeytabCredentialsService for keytab-based credential type
- Kerberos Principal + Kerberos Password: Component-level properties supporting password-based credentials
- Kerberos User Service: A property referencing the KerberosUserService controller service interface, with implementations for all credential types:
  - KerberosKeytabUserService for keytab-based credentials
  - KerberosPasswordUserService for password-based credentials
  - KerberosTicketCacheUserService for ticket cache-based credentials.

In NiFi 2, only the Kerberos User Service is retained because it can accommodate all credential types (keytab, password, ticket cache) with a single property on the component. The other configuration options have been removed in NiFi 2. For more information, see NIFI-13510.

Affected components in Cloudera Flow Management 2.1.7:

The tables below list the affected components and their legacy Kerberos properties, also indicating whether these components are available in Cloudera Flow Management 4.x.

Table 8: Cloudera Object Store processors

Component Type	Compo
org.apache.nifi.processors.hadoop.ListCDPObjectStore	nifi-cdf-
org.apache.nifi.processors.hadoop.FetchCDPObjectStore	nifi-cdf-

Component Type	Compo
org.apache.nifi.processors.hadoop.PutCDPObjectStore	nifi-cdf-
org.apache.nifi.processors.hadoop.DeleteCDPObjectStore	nifi-cdf-

Table 9: Hadoop processors

Component Type	Compo
org.apache.nifi.processors.hadoop.ListHDFS	nifi-hado
org.apache.nifi.processors.hadoop.FetchHDFS	nifi-hado
org.apache.nifi.processors.hadoop.PutHDFS	nifi-hado
org.apache.nifi.processors.hadoop.DeleteHDFS	nifi-hado
org.apache.nifi.processors.hadoop.GetHDFS	nifi-hado
org.apache.nifi.processors.hadoop.MoveHDFS	nifi-hado
org.apache.nifi.processors.hadoop.inotify.GetHDFSEvents	nifi-hado
org.apache.nifi.processors.hadoop.GetHDFSFileInfo	nifi-hado
org.apache.nifi.processors.hadoop.GetHDFSSequenceFile	nifi-hado
org.apache.nifi.processors.hadoop.CreateHadoopSequenceFile	nifi-hado
org.apache.nifi.processors.parquet.FetchParquet	nifi-parq
org.apache.nifi.processors.parquet.PutParquet	nifi-parq
org.apache.nifi.processors.orc.PutORC	nifi-hive

Table 10: HBase services

Component Type	Compo
org.apache.nifi.hbase.HBase_1_1_2_ClientService	nifi-hbas
org.apache.nifi.hbase.HBase_2_ClientService	nifi-hbas

Table 11: Hive components

Component Type	Compo
org.apache.nifi.dbcp.hive.HiveConnectionPool	nifi-hive
org.apache.nifi.dbcp.hive.Hive3ConnectionPool	nifi-hive
org.apache.nifi.processors.hive.PutHiveStreaming	nifi-hive
org.apache.nifi.processors.hive.PutHive3Streaming	nifi-hive

Table 12: Kafka processors

Component Type	Compo
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka_1_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafka_1_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord_1_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord_1_0	nifi-kafk

Component Type	Compo
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafka_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafka_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka2CDP	nifi-cdf-
org.apache.nifi.processors.kafka.pubsub.PublishKafka2CDP	nifi-cdf-
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord2CDP	nifi-cdf-
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord2CDP	nifi-cdf-

Table 13: Solr processors

Component Type	Compo
org.apache.nifi.processors.solr.GetSolr	nifi-solr-
org.apache.nifi.processors.solr.QuerySolr	nifi-solr-
org.apache.nifi.processors.solr.PutSolrContentStream	nifi-solr-
org.apache.nifi.processors.solr.PutSolrRecord	nifi-solr-

Table 14: Other processors

Component Type	Compo
org.apache.nifi.accumulo.controllerservices.AccumuloService	nifi-accu
org.apache.nifi.dbcp.DBCPConnectionPool	nifi-dbcp
org.apache.nifi.dbcp.HadoopDBCPConnectionPool	nifi-hado
org.apache.nifi.schemaregistry.hortonworks.HortonworksSchemaRegistry	nifi-hwx
org.apache.nifi.controller.kudu.KuduLookupService	nifi-kudu
org.apache.nifi.controller.livy.LivySessionController	nifi-livy-
org.apache.nifi.processors.kudu.PutKudu	nifi-kudu
org.apache.nifi.atlas.reporting.ReportLineageToAtlas	nifi-atlas

Flow migration for Kerberos configuration changes

To ensure compatibility with NiFi 2, you will need to migrate your flow by creating a new Kerberos UserService controller service based on the old controller service or component level properties. This process involves transitioning from legacy component-level properties to the updated service and clearing outdated configurations.

Migration steps for **Kerberos Principal + Kerberos Keytab**:

1. Create service: Set up a KerberosKeytabUserService using the existing Kerberos Principal and Kerberos Keytab component-level properties.
2. Update reference: Link the new service to the component's Kerberos User Service property.

3. Remove legacy properties: Clear the old property values from Kerberos Principal and Kerberos Keytab.

Migration steps for **Kerberos Credentials Service**:

1. Create service: Set up a KerberosKeytabUserService based on the properties from the KeytabCredentialsService.
2. Update reference: Point the component's Kerberos User Service property to the new service.
3. Remove legacy service: Clear the old Kerberos Credentials Service reference property and delete the outdated service.

Migration steps for **Kerberos Principal + Kerberos Password**:

1. Create service: Set up a KerberosPasswordUserService using the Kerberos Principal and Kerberos Password component-level properties.
2. Update reference: Link the new service to the component’s Kerberos User Service property.
3. Remove legacy properties: Clear the old property values from Kerberos Principal and Kerberos Password.

Components requiring initial code-level changes

For certain components, the Kerberos User Service property is not yet available. These components will require an initial code-level update before migration.

- Kafka_1_0 processors
- Kafka_2_0 processors
- CDPObjectStore processors
- ReportLineageToAtlas
- KuduLookupService
- LivySessionController

Migration to Cloudera-specific components

As part of the migration, some components will transition to new, Cloudera-specific types. The Kerberos property migration should be included in this transition.

- HortonworksSchemaRegistry => ClouderaSchemaRegistry
- Hive[3]ConnectionPool => ClouderaHiveConnectionPool
- PutHive[3]Streaming => PutClouderaHiveStreaming

Scripted components

In NiFi 2, support for certain languages in scripted components has been removed. The affected languages include ECMAScript, Lua, Ruby, and Python. Cloudera recommends to switch to Groovy or leverage the new Python API feature for developing processors. The following components are impacted by this change:

- org.apache.nifi.processors.script.ExecuteScript
- org.apache.nifi.processors.script.InvokeScriptedProcessor
- org.apache.nifi.processors.script.ScriptedFilterRecord
- org.apache.nifi.processors.script.ScriptedPartitionRecord
- org.apache.nifi.processors.script.ScriptedTransformRecord
- org.apache.nifi.processors.script.ScriptedValidateRecord
- org.apache.nifi.lookup.script.ScriptedLookupService
- org.apache.nifi.record.script.ScriptedReader
- org.apache.nifi.record.script.ScriptedRecordSetWriter
- org.apache.nifi.record.sink.script.ScriptedRecordSink
- org.apache.nifi.lookup.script.SimpleScriptedLookupService
- org.apache.nifi.reporting.script.ScriptedReportingTask

Important: As of now, the Python API in NiFi 2 is limited to creating processors. Writing controller services or reporting tasks using Python is not supported.

Custom components

If your NiFi environment includes custom components or NARs developed for NiFi 1.x, they are unlikely to be compatible with NiFi 2. To ensure compatibility, you must update your dependencies to align with NiFi 2 and rebuild your NARs using Java 21. This update is essential for a successful transition to the new version.

Flow controller level breaking changes

Transition from variables to parameters

Variables and the variable registry are removed in NiFi 2 due to their inherent limitations, such as requiring expression language support to reference a variable and the inability to store sensitive values. You can use parameter contexts instead, which have been significantly enhanced over recent years. For example, the addition of the Parameter Context Provider allows for sourcing parameter values from external stores (like HashiCorp Vault or cloud provider vaults).

This change is one of the most impactful in NiFi 2, which will require rework on existing data flows. However, it also presents an opportunity to optimize the organization of parameters, allowing you to split them into multiple parameter contexts and use inheritance when sharing parameters across different use cases.

Cloudera will provide automated tools within the NiFi Migration Tooling to assist with transitioning from variables to parameters.

Removal of XML templates

The concept of XML templates is being phased out in NiFi 2. Historically, these templates were stored in memory as well as in the flow definition files (flow.xml.gz and flow.json.gz). This caused significant issues for some NiFi users, especially those managing numerous large templates with thousands of components. Removing templates from NiFi will enhance stability and reduce memory usage.

If you use templates in your NiFi 1.x clusters, you should export your existing templates as JSON definitions or version them into a NiFi Registry instance to prepare for this change. Using NiFi Registry is the recommended best practice for version control, sharing, and reusing flow definitions.

If your template is a process group:

1. Drag and drop the template onto the canvas.
2. Right-click it, and choose to export it as a flow definition (JSON file) or start version control in your NiFi Registry, if you have one configured.

If your template is not a process group (just a flow with components):

1. Drag and drop a process group onto the canvas.
2. Go into that process group and drag and drop your template within that process group.
3. Go back to the parent process group containing your template and export it as a flow definition or start version control on it.

Cloudera will provide automated tools within the NiFi Migration Tooling to help manage the migration of templates.

Discontinuation of event driven thread pool

The Event Driven Scheduling Strategy, which was available for some processors in previous versions of NiFi, is being removed in NiFi 2. This feature was experimental and did not demonstrate significant performance improvements.

If you are using this scheduling strategy, you will need to update your components to use the time driven scheduling strategy instead. You can identify components using the Event Driven strategy by searching for “event” in the NiFi search bar.

NiFi framework breaking changes in NiFi 2

Java 21 compatibility

Java 21 is the minimum Java version required with NiFi 2. While most versions of NiFi 1.x may work with Java 21, it is not officially supported to run NiFi 1.x with Java 21 for extended periods in preparation for the upgrade. The transition to Java 21 should happen as part of the upgrade process to NiFi 2. You have to ensure that your environment is ready for Java 21 before initiating the upgrade.

Transition from flow.xml.gz to flow.json.gz

In NiFi 2, flow.xml.gz, which has been a cornerstone of flow configuration storage, is completely phased out and replaced by flow.json.gz. While the two files have coexisted in many NiFi 1.x releases, NiFi 2 will exclusively use JSON-based flow representations. Before upgrading, it is recommended to back up your flow.json.gz file. During the upgrade, only work with this JSON file if any updates are needed.

Repository encryption removal

The ability to encrypt repositories has been removed in NiFi 2.

NiFi Toolkit changes

The NiFi Toolkit will also undergo changes in NiFi 2, with some features being removed.

Dynamic parameter retrieval with parameter providers

In NiFi 2, the values associated with parameters retrieved by parameter providers are no longer stored within the flow.json.gz file. Instead, these values are retrieved on-demand or during NiFi's startup and are only stored in memory. This change enhances security and reduces the size of flow files, but it also means that parameter values will need to be available from their source each time NiFi is started or when the parameters are accessed. It is important to ensure that the external sources for these parameters are reliable and accessible to avoid disruptions during NiFi operations. For more information, see NIFI-13560.

Fixed issues in Cloudera Flow Management

Review the list of issues resolved in Cloudera Flow Management 4.11.0.

- CFM-5479: Added OAuth and Key-Pair authentication support for Snowflake controller services

  Snowflake controller services have been updated to support OAuth and key-pair authentication, replacing single-factor password authentication.
- CFM-5708: Pre upgrade check prevent NiFi 2.0 start after upgrade - unnecessary atlas reporting task check

  Fixed an issue where the upgrade pre-check script incorrectly prevented NiFi 2 from starting after an upgrade, due to an outdated check for the ReportLineageToAtlas reporting task. This component has been reintroduced and removed from the list of flagged issues. It is now recognized as a valid and permitted object.
- CFM-5737: Fixed PutIceberg processor failure caused by missing LiteralByteString class

  Fixed a compatibility issue where the PutIceberg processor failed when using Ozone as the file system. The issue was caused by missing dependencies after upgrading the protobuf library version in the Iceberg bundle.
- CFM-3829: Modified Cloudera Flow Management CSD to support managing both 2.x and 4.x versions with a single Cloudera Manager

  The Cloudera Flow Management Custom Service Descriptor (CSD) has been updated to allow you to install and manage both 2.x and 4.x versions in parallel using a single Cloudera Manager instance. This enhancement

addresses use cases where you need to operate multiple clusters with different Cloudera Flow Management versions. It allows you to run, test, and migrate flows between NiFi 1 and NiFi 2 clusters without requiring separate Cloudera Manager deployments, reducing administrative overhead and simplifying the migration process.

Fixed Common Vulnerabilities and Exposures in Cloudera Flow Management in 4.11.0

CVE-2014-3488

The SslHandler in Netty before 3.9.2 allows remote attackers to cause a denial of service (infinite loop and CPU consumption) via a crafted SSLv2Hello message.

CVE-2015-2156

Netty before 3.9.8.Final, 3.10.x before 3.10.3.Final, 4.0.x before 4.0.28.Final, and 4.1.x before 4.1.0.Beta5 and Play Framework 2.x before 2.3.9 might allow remote attackers to bypass the httpOnly flag on cookies and obtain sensitive information by leveraging improper validation of cookie name and value characters.

CVE-2016-6811

In Apache Hadoop 2.x before 2.7.4, a user who can escalate to yarn user can possibly run arbitrary commands as root user.

CVE-2017-15713

Vulnerability in Apache Hadoop 0.23.x, 2.x before 2.7.5, 2.8.x before 2.8.3, and 3.0.0-alpha through 3.0.0-beta1 allows a cluster user to expose private files owned by the user running the MapReduce job history server process.

The malicious user can construct a configuration file containing XML directives that reference sensitive files on the MapReduce job history server host.

CVE-2017-18640

The Alias feature in SnakeYAML before 1.26 allows entity expansion during a load operation, a related issue to CVE-2003-1564.

CVE-2017-3161

The HDFS web Ul in Apache Hadoop before 2.7.0 is vulnerable to a cross-site scripting (XSS) attack through an unescaped query parameter.

CVE-2017-3162

HDFS clients interact with a servlet on the DataNode to browse the HDFS namespace. The NameNode is provided as a query parameter that is not validated in Apache Hadoop before 2.7.0.

CVE-2018-11768

In Apache Hadoop 3.1.0 to 3.1.1, 3.0.0-alpha1 to 3.0.3, 2.9.0 to 2.9.1, and 2.0.0-alpha to 2.8.4, the user/group information can be corrupted across storing in fsimage and reading back from fsimage.

CVE-2018-8029

In Apache Hadoop versions 3.0.0-alpha1 to 3.1.0, 2.9.0 to 2.9.1, and 2.2.0 to 2.8.4, a user who can escalate to yarn user can possibly run arbitrary commands as root user.

CVE-2019-16869

Netty before 4.1.42.Final mishandles whitespace before the colon in HTTP headers (such as a "Transfer-Encoding: chunked" line), which leads to HTTP request smuggling.

CVE-2019-20444

HttpObjectDecoder.java in Netty before 4.1.44 allows an HTTP header that lacks a colon, which might be interpreted as a separate header with an incorrect syntax, or might be interpreted as an "invalid fold."

CVE-2019-20445

HttpObjectDecoder.java in Netty before 4.1.44 allows a Content-Length header to be accompanied by a second Content-Length header, or by a Transfer-Encoding header.

CVE-2020-11612

The ZlibDecoders in Netty 4.1.x before 4.1.46 allow for unbounded memory allocation while decoding a ZlibEncoded byte stream.

An attacker could send a large ZlibEncoded byte stream to the Netty server, forcing the server to allocate all of its free memory to a single decoder.

CVE-2020-9040

Couchbase Server Java SDK before 2.7.1.1 allows a potential attacker to forge an SSL certificate and pose as the intended peer.

An attacker can leverage this flaw by crafting a cryptographically valid certificate that will be accepted by Java SDK's Netty component due to missing hostname verification.

CVE-2021-0341

In verifyHostName of OkHostnameVerifier.java, there is a possible way to accept a certificate for the wrong domain due to improperly used crypto.

This could lead to remote information disclosure with no additional execution privileges needed. User interaction is not needed for exploitation.Product: Android Versions: Android-8.1 Android-9 Android-10 Android-11Android ID: A-171980069

CVE-2021-21290: Local Information Disclosure Vulnerability in Netty on Unix-Like systems due temporary files

Netty is an open-source, asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients. In Netty before version 4.1.59.Final there is a vulnerability on Unix-like systems involving an insecure temp file.

When netty's multipart decoders are used local information disclosure can occur via the local system temporary directory if temporary storing uploads on the disk is enabled. On unix-like systems, the temporary directory is shared between all user.

As such, writing to this directory using APIs that do not explicitly set the file/directory permissions can lead to information disclosure. Of note, this does not impact modern MacOS Operating Systems.

The method "File.createTempFile" on unix-like systems creates a random file, but, by default will create this file with the permissions "-rw-r--r--". Thus, if sensitive information is written to this file, other local users can read this information. This is the case in netty's "AbstractDiskHttpData" is vulnerable. This has been fixed in version 4.1.59.Final.

As a workaround, one may specify your own "java.io.tmpdir" when you start the JVM or use "DefaultHttpDataFactory.setBaseDir(...)" to set the directory to something that is only readable by the current user.

CVE-2021-21295: Possible request smuggling in HTTP/2 due missing validation

Netty is an open-source, asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients. In Netty (io.netty:netty-codec-http2) before version 4.1.60.Final there is a vulnerability that enables request smuggling.

If a Content-Length header is present in the original HTTP/2 request, the field is not validated by 'Http2MultiplexHandler' as it is propagated up. This is fine as long as the request is not proxied through as HTTP/1.1.

If the request comes in as an HTTP/2 stream, gets converted into the HTTP/1.1 domain objects (`HttpRequest`, `HttpContent`, etc.) via `Http2StreamFrameToHttpObjectCodec` and then sent up to the child channel's pipeline and proxied through a remote peer as HTTP/1.1 this may result in request smuggling.

In a proxy case, users may assume the content-length is validated somehow, which is not the case. If the request is forwarded to a backend channel that is a HTTP/1.1 connection, the Content-Length now has meaning and needs to be checked. An attacker can smuggle requests inside the body as it gets downgraded from HTTP/2 to HTTP/1.1. For an example attack refer to the linked GitHub Advisory.

Users are only affected if all of this is true: 'HTTP2MultiplexCodec or Http2FrameCodec' is used, 'Http2StreamFrameToHttpObjectCodec' is used to convert to HTTP/1.1 objects, and these HTTP/1.1 objects are forwarded to another remote peer.

This has been patched in 4.1.60.Final. As a workaround, the user can do the validation by themselves by implementing a custom 'ChannelInboundHandler' that is put in the 'ChannelPipeline' behind `Http2StreamFrameToHttpObjectCodec`.

CVE-2021-21409: Possible request smuggling in HTTP/2 due missing validation of content-length

Netty is an open-source, asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients. In Netty (io.netty:netty-codec-http2) before version 4.1.61.Final there is a vulnerability that enables request smuggling.

The content-length header is not correctly validated if the request only uses a single Http2HeaderFrame with the endStream set to true. This could lead to request smuggling if the request is proxied to a remote peer and translated to HTTP/1.1. This is a followup of GHSA-wm47-8v5p-wjpj/CVE-2021-21295 which did miss to fix this one case. This was fixed as part of 4.1.61.Final.

CVE-2021-34371

Neo4j through 3.4.18 (with the shell server enabled) exposes an RMI service that arbitrarily deserializes Java objects, e.g., through setSessionVariable.

An attacker can abuse this for remote code execution because there are dependencies with exploitable gadget chains.

CVE-2021-37136

The Bzip2 decompression decoder function doesn't allow setting size restrictions on the decompressed output data (which affects the allocation size used during decompression).

All users of Bzip2Decoder are affected. The malicious input can trigger an OOME and so a DoS attack.

CVE-2021-37137

The Snappy frame decoder function doesn't restrict the chunk length which may lead to excessive memory usage.

Beside this it also may buffer reserved skippable chunks until the whole chunk was received which may lead to excessive memory usage as well.

This vulnerability can be triggered by supplying malicious input that decompresses to a very big size (via a network stream or a file) or by sending a huge skippable chunk.

CVE-2021-37533: Apache Commons Net's FTP client trusts the host from PASV response by default

Prior to Apache Commons Net 3.9.0, Net's FTP client trusts the host from PASV response by default.

A malicious server can redirect the Commons Net code to use a different host, but the user has to connect to the malicious server in the first place.

This may lead to leakage of information about services running on the private network of the client. The default in version 3.9.0 is now false to ignore such hosts, as cURL does. See https://issues.apache.org/jira/browse/NET-711.

CVE-2021-43797: HTTP fails to validate against control chars in header names which may lead to HTTP request smuggling

Netty is an asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients. Netty prior to version 4.1.71.Final skips control chars when they are present at the beginning / end of the header name.

It should instead fail fast as these are not allowed by the spec and could lead to HTTP request smuggling.

Failing to do the validation might cause netty to "sanitize" header names before it forward these to another remote system when used as proxy. This remote system can't see the invalid usage anymore, and therefore does not do the validation itself.

Users should upgrade to version 4.1.71.Final.

CVE-2022-1471: Remote Code execution in Snake YAML

SnakeYaml's Constructor() class does not restrict types which can be instantiated during deserialization.

Deserializing yaml content provided by an attacker can lead to remote code execution.

We recommend using SnakeYaml's SafeConsturctor when parsing untrusted content to restrict deserialization. We recommend upgrading to version 2.0 and beyond.

CVE-2022-24823: Local Information Disclosure Vulnerability in io.netty:netty-codec-http

Netty is an open-source, asynchronous event-driven network application framework. The package io.netty:netty-codec-http prior to version 4.1.77.Final contains an insufficient fix for CVE-2021-21290.

When Netty's multipart decoders are used local information disclosure can occur via the local system temporary directory if temporary storing uploads on the disk is enabled.

This only impacts applications running on Java version 6 and lower. Additionally, this vulnerability impacts code running on Unix-like systems, and very old versions of Mac OSX and Windows as they all share the system temporary directory between all users.

Version 4.1.77.Final contains a patch for this vulnerability. As a workaround, specify one's own java.io.tmpdir when starting the JVM or use DefaultHttpDataFactory.setBaseDir(...) to set the directory to something that is only readable by the current user.

CVE-2022-25857: Denial of Service (DoS)

The package org.yaml:snakeyaml from 0 and before 1.31 are vulnerable to Denial of Service (DoS) due missing to nested depth limitation for collections.

CVE-2022-30187: Azure Storage Library Information Disclosure Vulnerability

Azure Storage Library Information Disclosure Vulnerability

CVE-2022-38749: DoS in SnakeYAML

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS).

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

CVE-2022-38750: DoS in SnakeYAML

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS).

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

CVE-2022-38751: DoS in SnakeYAML

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS).

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow.

CVE-2022-38752: DoS in SnakeYAML

Using snakeYAML to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS).

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack-overflow.

CVE-2022-40149: Stack Buffer Overflow in Jettison

Those using Jettison to parse untrusted XML or JSON data may be vulnerable to Denial of Service attacks (DOS).

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow. This effect may support a denial of service attack.

CVE-2022-40150: Stack Buffer Overflow in Jettison

Those using Jettison to parse untrusted XML or JSON data may be vulnerable to Denial of Service attacks (DOS).

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by Out of memory. This effect may support a denial of service attack.

CVE-2022-40152: Stack Buffer Overflow in Woodstox

Those using Woodstox to parse XML data may be vulnerable to Denial of Service attacks (DOS) if DTD support is enabled.

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stackoverflow. This effect may support a denial of service attack.

CVE-2022-41854: Stack Overflow in Snakeyaml

Those using Snakeyaml to parse untrusted YAML files may be vulnerable to Denial of Service attacks (DOS).

If the parser is running on user supplied input, an attacker may supply content that causes the parser to crash by stack overflow. This effect may support a denial of service attack.

CVE-2022-41881

Netty project is an event-driven asynchronous network application framework. In versions prior to 4.1.86.Final, a StackOverflowError can be raised when parsing a malformed crafted message due to an infinite recursion.

This issue is patched in version 4.1.86.Final. There is no workaround, except using a custom HaProxyMessageDecoder.

CVE-2022-45685

A stack overflow in Jettison before v1.5.2 allows attackers to cause a Denial of Service (DoS) via crafted JSON data.

CVE-2022-45693

Jettison before v1.5.2 was discovered to contain a stack overflow via the map parameter. This vulnerability allows attackers to cause a Denial of Service (DoS) via a crafted string.

CVE-2023-0833: Red hat a-mq streams: component version with information disclosure flaw

A flaw was found in Red Hat's AMQ-Streams, which ships a version of the OKHttp component with an information disclosure flaw via an exception triggered by a header containing an illegal value.

This issue could allow an authenticated attacker to access information outside of their regular permissions.

CVE-2023-1436: Infinite recursion in Jettison leads to denial of service when creating a crafted JSONArray

An infinite recursion is triggered in Jettison when constructing a JSONArray from a Collection that contains a self-reference in one of its elements.

This leads to a StackOverflowError exception being thrown.

CVE-2023-20860

Spring Framework running version 6.0.0 - 6.0.6 or 5.3.0-5.3.25 using as a pattern in Spring Security configuration with the mvcRequestMatcher creates a mismatch in pattern matching between Spring Security and Spring MVC, and the potential for a security bypass.

CVE-2023-20861

In Spring Framework versions 6.0.0 - 6.0.6, 5.3.0-5.3.25, 5.2.0.RELEASE - 5.2.22.RELEASE, and older unsupported versions, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition.

CVE-2023-20863

In spring framework versions prior to 5.2.24 release+, 5.3.27+ and 6.0.8+, it is possible for a user to provide a specially crafted SpEL expression that may cause a denial-of-service (DoS) condition.

CVE-2023-34042

The `spring-security.xsd` file inside the `spring-security-config` jar is world writable which means that if it were extracted it could be written by anyone with access to the file system.

While there are no known exploits, this is an example of "CWE-732: Incorrect Permission Assignment for Critical Resource" and could result in an exploit.

Users should update to the latest version of Spring Security to mitigate any future exploits found around this issue.

CVE-2023-34453: snappy-java's Integer Overflow vulnerability in shuffle leads to DoS

snappy-java is a fast compressor/decompressor for Java. Due to unchecked multiplications, an integer overflow may occur in versions prior to 1.1.10.1, causing a fatal erro#.

The function `shuffle(int[] input)` in the file `BitShuffle.java` receives an array of integers and applies a bit shuffle on it. It does so by multiplying the length by 4 and passing it to the natively compiled shuffle function.

Since the length is not tested, the multiplication by four can cause an integer overflow and become a smaller value than the true size, or even zero or negative.

In the case of a negative value, a `java.lang.NegativeArraySizeException` exception will raise, which can crash the program. In a case of a value that is zero or too small, the code that afterwards references the shuffled array will assume a bigger size of the array, which might cause exceptions such as `java.lang.ArrayIndexOutOfBoundsException`.

The same issue exists also when using the `shuffle` functions that receive a double, float, long and short, each using a different multiplier that may cause the same issue.

Version 1.1.10.1 contains a patch for this vulnerability.

CVE-2023-34454: snappy-java's Integer Overflow vulnerability in compress leads to DoS

snappy-java is a fast compressor/decompressor for Java. Due to unchecked multiplications, an integer overflow may occur in versions prior to 1.1.10.1, causing an unrecoverable fatal error.

The function `compress(char[] input)` in the file `Snappy.java` receives an array of characters and compresses it. It does so by multiplying the length by 2 and passing it to the rawCompress function.

Since the length is not tested, the multiplication by two can cause an integer overflow and become negative.

The rawCompress function then uses the received length and passes it to the natively compiled maxCompressedLength function, using the returned value to allocate a byte array.

Since the maxCompressedLength function treats the length as an unsigned integer, it doesn't care that it is negative, and it returns a valid value, which is casted to a signed integer by the Java engine.

If the result is negative, a `java.lang.NegativeArraySizeException` exception will be raised while trying to allocate the array `buf`. On the other side, if the result is positive, the `buf` array will successfully be allocated, but its size might be too small to use for the compression, causing a fatal Access Violation error.

The same issue exists also when using the `compress` functions that receive double, float, int, long and short, each using a different multiplier that may cause the same issue. The issue most likely won't occur when using a byte array, since creating a byte array of size 0×80000000 (or any other negative value) is impossible in the first place.

Version 1.1.10.1 contains a patch for this issue.

CVE-2023-34455: snappy-java's unchecked chunk length leads to DoS

snappy-java is a fast compressor/decompressor for Java. Due to use of an unchecked chunk length, an unrecoverable fatal error can occur in versions prior to 1.1.10.1.

The code in the function `hasNextChunk` in the file `SnappyInputStream.java` checks if a given stream has more chunks to read. It does that by attempting to read 4 bytes. If it wasn't possible to read the 4 bytes, the function returns false. Otherwise, if 4 bytes were available, the code treats them as the length of the next chunk.

In the case that the `compressed` variable is null, a byte array is allocated with the size given by the input data.

Since the code doesn't test the legality of the `chunkSize` variable, it is possible to pass a negative number (such as 0xFFFFFFFF which is -1), which will cause the code to raise a `java.lang.NegativeArraySizeException` exception. A worse case would happen when passing a huge positive value (such as 0x7FFFFFFF), which would raise the fatal `java.lang.OutOfMemoryError` error.

Version 1.1.10.1 contains a patch for this issue.

CVE-2023-34462: netty-handler SniHandler 16MB allocation

Netty is an asynchronous event-driven network application framework for rapid development of maintainable high performance protocol servers & clients.

The SniHandler can allocate up to 16MB of heap for each channel during the TLS handshake.

When the handler or the channel does not have an idle timeout, it can be used to make a TCP server using the `SniHandler` to allocate 16MB of heap.

The `SniHandler` class is a handler that waits for the TLS handshake to configure a `SslHandler` according to the indicated server name by the ClientHello record. For this matter it allocates a `ByteBuf` using the value defined in the ClientHello record.

Normally the value of the packet should be smaller than the handshake packet but there are not checks done here and the way the code is written, it is possible to craft a packet that makes the SslClientHelloHandler.

This vulnerability has been fixed in version 4.1.94.Final.

CVE-2023-36415: Azure Identity SDK Remote Code Execution Vulnerability

Azure Identity SDK Remote Code Execution Vulnerability

CVE-2023-39410: Apache Avro Java SDK: Memory when deserializing untrusted data in Avro Java SDK

When deserializing untrusted or corrupted data, it is possible for a reader to consume memory beyond the allowed constraints and thus lead to out of memory on the system.

This issue affects Java applications using Apache Avro Java SDK up to and including 1.11.2.

Users should update to apache-avro version 1.11.3 which addresses this issue.

CVE-2023-43642: Missing upper bound check on chunk length in snappy-java

snappy-java is a Java port of the snappy, a fast C++ compresser/decompresser developed by Google.

The SnappyInputStream was found to be vulnerable to Denial of Service (DoS) attacks when decompressing data with a too large chunk size.

Due to missing upper bound check on chunk length, an unrecoverable fatal error can occur.

All versions of snappy-java including the latest released version 1.1.10.3 are vulnerable to this issue.

A fix has been introduced in commit 9f8c3cf74 which will be included in the 1.1.10.4 release. Users are advised to upgrade. Users unable to upgrade should only accept compressed data from trusted sources.

CVE-2023-44487

The HTTP/2 protocol allows a denial of service (server resource consumption) because request cancellation can reset many streams quickly, as exploited in the wild in August through October 2023.

CVE-2023-52428

In Connect2id Nimbus JOSE+JWT before 9.37.2, an attacker can cause a denial of service (resource consumption) via a large JWE p2c header value (aka iteration count) for the PasswordBased Decrypter (PBKDF2) component.

CVE-2024-13009: Eclipse Jetty GZIP buffer release

In Eclipse Jetty versions 9.4.0 to 9.4.56 a buffer can be incorrectly released when confronted with a gzip error when inflating a request body.

This can result in corrupted and/or inadvertent sharing of data between requests.

CVE-2024-22257

In Spring Security, versions 5.7.x prior to 5.7.12, 5.8.x prior to 5.8.11, versions 6.0.x prior to 6.0.9, versions 6.1.x prior to 6.1.8, versions 6.2.x prior to 6.2.3, an application is possible vulnerable to broken access control when it directly uses the `AuthenticatedVoter#vote` passing a null Authentication parameter.

CVE-2024-29131: Apache Commons Configuration: StackOverflowError adding property in AbstractListDelimiterHandler.flattenIterator()

Out-of-bounds Write vulnerability in Apache Commons Configuration. This issue affects Apache Commons Configuration: from 2.0 before 2.10.1.

Users are recommended to upgrade to version 2.10.1, which fixes the issue.

CVE-2024-29133: Apache Commons Configuration: StackOverflowError calling ListDelimiterHandler.flatten (Object, int) with a cyclical object tree

Out-of-bounds Write vulnerability in Apache Commons Configuration. This issue affects Apache Commons Configuration: from 2.0 before 2.10.1.

Users are recommended to upgrade to version 2.10.1, which fixes the issue.

CVE-2024-35255: Azure Identity Libraries and Microsoft Authentication Library Elevation of Privilege Vulnerability

Azure Identity Libraries and Microsoft Authentication Library Elevation of Privilege Vulnerability

CVE-2024-38820: Spring Framework DataBinder Case Sensitive Match Exception

The fix for CVE-2022-22968 made disallowed Fields patterns in DataBinder case insensitive.

However, String.toLowerCase() has some Locale dependent exceptions that could potentially result in fields not protected as expected.

CVE-2024-43591: Azure Command Line Integration (CLI) Elevation of Privilege Vulnerability

Azure Command Line Integration (CLI) Elevation of Privilege Vulnerability

CVE-2024-47561: Apache Avro Java SDK: Arbitrary Code Execution when reading Avro schema (Java SDK)

Schema parsing in the Java SDK of Apache Avro 1.11.3 and previous versions allows bad actors to execute arbitrary code.

Users are recommended to upgrade to version 1.11.4 or 1.12.0, which fix this issue.

CVE-2024-6763: Jetty URI parsing of invalid authority

Eclipse Jetty is a lightweight, highly scalable, Java-based web server and Servlet engine. It includes a utility class, HttpURI, for URI/URL parsing.

The HttpURI class does insufficient validation on the authority segment of a URI. However the behaviour of HttpURI differs from the common browsers in how it handles a URI that would be considered invalid if fully validated against the RRC.

Specifically HttpURI and the browser may differ on the value of the host extracted from an invalid URI and thus a combination of Jetty and a vulnerable browser may be vulnerable to a open redirect attack or to a SSRF attack if the URI is used after passing validation checks.

CVE-2024-8184: Jetty `ThreadLimitHandler.getRemote()` vulnerable to remote DoS attacks

There exists a security vulnerability in Jetty's `ThreadLimitHandler.getRemote()` which can be exploited by unauthorized users to cause remote denial-of-service (DoS) attack.

By repeatedly sending crafted requests, attackers can trigger OutofMemory errors and exhaust the server's memory.

CVE-2025-22228: Spring Security BCryptPasswordEncoder does not enforce maximum password length

`BCryptPasswordEncoder.matches(CharSequence, String)` will incorrectly return true for passwords larger than 72 characters as long as the first 72 characters are the same.

CVE-2025-25193: Denial of Service attack on windows app using Netty

Netty, an asynchronous, event-driven network application framework, has a vulnerability in versions up to and including 4.1.118.Final. An unsafe reading of environment file could potentially cause a denial of service in Netty. When loaded on an Windows application, Netty attempts to load a file that does not exist. If an attacker creates such a large file, the Netty application crash. A similar issue was previously reported as CVE-2024-47535.

This issue was fixed, but the fix was incomplete in that null-bytes were not counted against the input limit. Commit d1fbda62d3a47835d3fb35db8bd42ecc205a5386 contains an updated fix.

CVE-2025-27553: Apache Commons VFS: Possible path traversal issue when using NameScope.DESCENDENT

Relative Path Traversal vulnerability in Apache Commons VFS before 2.10.0.

The FileObject API in Commons VFS has a 'resolveFile' method that takes a 'scope' parameter.

Specifying `NameScope.DESCENDENT` promises that "an exception is thrown if the resolved file is not a descendent of the base file".

However, when the path contains encoded ".." characters (for example, "%2E%2E/bar.txt"), it might return file objects that are not a descendent of the base file, without throwing an exception.

This issue affects Apache Commons VFS: before 2.10.0. Users are recommended to upgrade to version 2.10.0, which fixes the issue.

CVE-2025-41232: Spring Security authorization bypass for method security annotations on private methods

Spring Security Aspects may not correctly locate method security annotations on private methods. This can cause an authorization bypass.

Your application may be affected by this if the following are true: You are using `@EnableMethodSecurity(mode=ASPECTJ)` and `spring-security-aspects`, and you have Spring Security method annotations on a private method.

In that case, the target method may be able to be invoked without proper authorization.

You are not affected if: You are not using `@EnableMethodSecurity(mode=ASPECTJ)` or `spring-security-aspects`, or You have no Spring Security-annotated private methods.

CVE-2025-41234: RFD Attack via "Content-Disposition" Header Sourced from Request

In Spring Framework, versions 6.0.x as of 6.0.5, versions 6.1.x and 6.2.x, an application is vulnerable to a reflected file download (RFD) attack when it sets a "Content-Disposition" header with a non-ASCII charset, where the filename attribute is derived from user-supplied input.

Specifically, an application is vulnerable when all the following are true: The header is prepared with `org.springframework.http.ContentDisposition`.

The filename is set via `ContentDisposition.Builder#filename(String, Charset)`. The value for the filename is derived from user-supplied input. The application does not sanitize the user-supplied input. The downloaded content of the response is injected with malicious commands by the attacker (see RFD paper reference for details).

An application is not vulnerable if any of the following is true: The application does not set a "Content-Disposition" response header. The header is not prepared with `org.springframework.http.ContentDisposition`. The filename is set via one of: `ContentDisposition.Builder#filename(String)`, or `ContentDisposition.Builder#filename(String, ASCII)`. The filename is not derived from user-supplied input. The filename is derived from user-supplied input but sanitized by the application. The attacker cannot inject malicious content in the downloaded content of the response.

Affected Spring Products and VersionsSpring Framework: 6.2.0-6.2.7, 6.1.0-6.1.20, 6.0.5-6.0.28. Older, unsupported versions are not affected.

Mitigation: Users of affected versions should upgrade to the corresponding fixed version.

Affected version(s) Fix versionAvailability: 6.2.x to 6.2.8 OSS, 6.1.x to 6.1.21 OSS, 6.0.x to 6.0.29 Commercial https://enterprise.spring.io/ No further mitigation steps are necessary.

CWE-113 in `Content-Disposition` handling in VMware Spring Framework versions 6.0.5 to 6.2.7 allows remote attackers to launch Reflected File Download (RFD) attacks via unsanitized user input in 'ContentDisposition.Builder#filename(String, Charset) with non-ASCII charsets.

CVE-2025-48924: Apache Commons Lang, Apache Commons Lang: `ClassUtils.getClass(...)` can throw a StackOverflowError on very long inputs

Uncontrolled Recursion vulnerability in Apache Commons Lang.

This issue affects Apache Commons Lang: Starting with `commons-lang:commons-lang` 2.0 to 2.6, and, from `org.apache.commons:commons-lang3` 3.0 before 3.18.0.

The methods `ClassUtils.getClass(...)` can throw `StackOverflowError` on very long inputs.

Because an Error is usually not handled by applications and libraries, a `StackOverflowError` could cause an application to stop.

Users are recommended to upgrade to version 3.18.0, which fixes the issue.

CVE-2025-48976: Apache Commons FileUpload, Apache Commons FileUpload: FileUpload DoS via part headers

Allocation of resources for multipart headers with insufficient limits enabled a DoS vulnerability in Apache Commons FileUpload.

This issue affects Apache Commons FileUpload: from 1.0 before 1.6; from 2.0.0-M1 before 2.0.0-M4.

Users are recommended to upgrade to versions 1.6 or 2.0.0-M4, which fix the issue.