LocationProperties

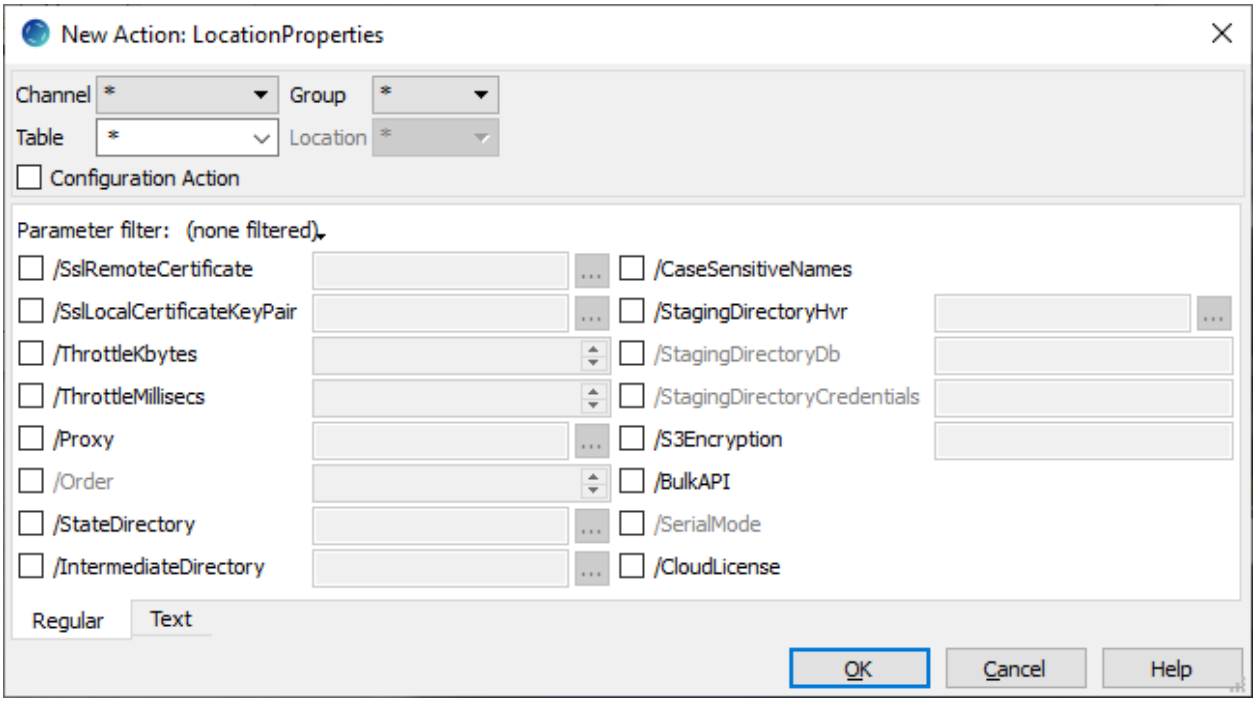
Contents

Description

Action **LocationProperties** defines properties of a remote location. This action has no affect other than that of its parameters. If this action is defined on a specific table, then it affects the entire job including data from other tables for that location.

Parameters

This section describes the parameters available for action **LocationProperties**. By default, only the supported parameters available for the selected location class are displayed in the **LocationProperties** win dow.



Parameter	Argument	Description
/SslRemoteCertificate	*pubcert*	Enable Secure Socket Layer (SSL) network encryption and check the identity of a remote location using *pubcert* file. Encryption relies on a public certificate which is held on the hub and remote location and a corresponding private key which is only held on the remote location. New pairs of private key and public certificate files can be generated by command **hvrsslgen** and are supplied to the remote **hvr** executable or **Hvrremotelistener** service with option **–K**. The argument *pubcert* points to the public certificate of the remote location which should be visible on the hub machine. It should either be an absolute pathname or a relative pathname (HVR then looks in directory **$HVR_HOME/lib**). A typical value is **hvr** which refers to a standard public certificate **$HVR_HOME/lib/cert/hvr.pub_cert.**

/SslLocalCertificateKeyPair	*pair*	Enable Secure Socket Layer (SSL) network encryption and allow the remote location to check the hub's identity by matching its copy of the hub's public certificate against *pair* which points to the hub machine's private key and public certificate pair. New pairs of private key and public certificate files can be generated by command **hvrsslgen** and are supplied to the remote **hvr** executable or **hvrremotelistener** service using an XML file containing the HVR access list. The argument *pair* points to the public certificate of the remote location which should be visible on the hub machine. It should either be an absolute pathname or a relative pathname (HVR then looks in directory **$HVR_HOME/lib**). It specifies two files: the names of these files are calculated by removing any extension from *pair* and then adding extensions **.pub_cert** and **.priv_key**. For example, value **hvr** refers to files **$HVR_HOME/lib/cert/hvr.pub_cert** and **$HVR_HOME/lib/cert/hvr.priv_key.**
/ThrottleKbytes	*int*	Restrain network bandwidth usage by grouping data sent to/from remote connection into packages, each containing *int* bytes, followed by a short sleep. The duration of the sleep is defined by **/ThrottleMillisecs**. Carefully setting these parameters will prevent HVR being an 'anti–social hog' of precious network bandwidth. This means it will not interfere with interactive end–users who share the link for example. For example if a network link can handle 64 KB /sec then a throttle of 32 KB with a 500 millisecond sleep will ensure HVR would be limited to no more than 50% bandwidth usage (when averaged–out over a one second interval).While using this parameter ensure to provide value in the dependent parameter **/ThrottleMillisecs**.
/ThrottleMillisecs	*int*	Restrict network bandwidth usage by sleeping *int* milliseconds between packets. While using this parameter ensure to provide value in the dependent parameter **/ThrottleKbytes** to define the package size.
/Proxy	*url*	URL of proxy server to connect to the specific location. Proxy servers are supported when connecting to HVR remote locations, for remote file access protocols (FTP, SFTP, WebDAV) and for Salesforce locations. The proxy server will be used for connections from the hub machine. If a remote HVR location is defined, then HVR will connect using its own protocol to the HVR remote machine and then via the proxy to the FTP/SFTP/WebDAV/Salesforce machine. Example, value *url* can be **hvr://***host name***:***port number*
/Order	*N*	Specify order of proxy chain from hub to location. Proxy chaining is only supported to HVR remote locations, not for file proxies (FTP, SFTP, WebDAV) or Salesforce proxies.

/StateDirectory	*path*	Directory for internal files used by HVR file replication state. By **default** these files are created in subdirectory **_hvr_state** which is created inside the file location top directory. If *path* is relative (e.g. **../work**), then the path used is relative to the file location's top directory. The state directory can either be defined to be a path inside the location's top directory or put outside this top directory. If the state directory is on the same file system as the file location's top directory, then HVR integrates file move operations will be 'atomic', so users will not be able to see the file partially written. Defining this parameter on a SharePoint /WebDAV integrate location ensures that the SharePoint version history is preserved.
/IntermediateDirectory **Since** v5.5.5/6	*dir*	Directory for storing 'intermediate files' that are generated during compare. Intermediate files are generated by file 'pre-read' subtasks while performing direct file compare. If this parameter is not defined, then by **default** the intermediate files are stored in *integratedir*/**_hvr_intermediate** directory. The *integratedir* is the replication **Directory** defined in the **New Location** screen while creating a file location.
/CaseSensitiveNames		DBMS table names and column names are treated case sensitive by HVR. Normally HVR converts table names to lowercase and treats table and column names as case insensitive. Settings this parameter allows the replication of tables with mixed case names or tables whose names do not match the DBMS case convention. For example, normally an Oracle table name is held in uppercase internally (e.g. **MYTAB**), so this parameter is needed to replicate a table named **mytab** or **MyTab**. This parameter is supported only for certain location classes. For the list of location classes that supports this parameter, see Treat DBMS table names and columns case sensitive in Capabilities.
/StagingDirectoryHvr	*URL*	Directory for bulk load staging files. For certain databases (Hive ACID, Redshift, and Snowflake), HVR splits large amount data into multiple staging files, to optimize performance. This parameter is supported only for certain location classes. For the list of supported location classes, see Bulk load requires a staging area in Capabilities. For Hive ACID, this should be an S3 or HDFS location.For Greenplum and HANA, this should be a local directory on the machine where HVR connects to the DBMS. For MySQL/MariaDb, when direct loading by the MySQL/MariaDB server option is used, this should be a directory local to the MySQL /MariaDB server which can be written to by the HVR user from the machine that HVR uses to connect to the DBMS. And when initial loading by the MySQL/MariaDB client option is used, this should be a local directory on the machine where HVR connects to the DBMS. For Redshift and Snowflake, this should be an S3 location.

/StagingDirectoryDb	*URL*	Location for the bulk load staging files visible from the database. This should point to the same files as **/StagingDirectoryHvr**. This parameter is enabled only if **/StagingDirectoryHvr** is selected. This parameter is supported only for certain location classes. For the list of supported location classes, see Bulk load requires a staging area in Capabilities. For Greenplum, this should either be a local directory on the Greenplum head-node or it should be a URL pointing to **/StagingDirectoryHvr**, for example a path starting with **gpfdist:** or **gpfdists:**. For HANA, this should be a local directory on the HANA machine which is configured for importing data by HANA. For Hive ACID, this should be the S3 or HDFS location that is used for **StagingDirectoryHvr**. For MySQL/MariaDb, when direct loading by the MySQL/MariaDB server option is used, this should be the directory from which the MySQL/MariaDB server should load the files. And when initial loading by the MySQL/MariaDB client option is used, this should be left empty. For Redshift and Snowflake, this should be the S3 location that is used for **StagingDirectoryHvr**.
/StagingDirectoryCredentials	*credentials*	Credentials to be used for S3 authentication and optional encryption during Hive ACID, RedShift, and Snowflake bulk load. This parameter is enabled only if **/StagingDirectoryDb** is selected. This parameter is supported only for certain location classes. For the list of supported location classes, see Bulk load requires a staging area in Capabilities. The supported formats for providing the credentials are: • Use credentials to retrieve from the AWS or Azure console, For AWS, **aws_access_key_id=***access_key_id***; aws_secret_access_key=***secret_access_key***;** For Azure, **azure_account=***azure_account***;azure_secret_access_key=***secret_key***;** • Use temporary credentials from the role of the current AWS EC2 node, **role=***AWS_IAM_role_name***;** • Optionally a static symmetric key for AES256 encryption of staged files can be provided with the above options for AWS, **aws_access_key_id=***access_key_id***; aws_secret_access_key=***secret_access_key***; master_symmetric_key=***32_hex_digits***;** or **role=***AWS_IAM_role_name***;master_symmetric_key=***32_hex_digits***;**
/S3Encryption	*keyinfo*	Enable client or server side encryption for uploading files into S3 locations. When client side encryption is enabled, any file uploaded to S3 is encrypted by HVR prior to uploading. With server side encryption, files uploaded to S3 will be encrypted by the S3 service itself. Value *keyinfo* can be:

- **sse_s3**
- **sse_kms**
- **master_symmetric_key=***64_hex_digits*
- **kms_cmk_id=***aws_kms_key_identifier*
  If only **kms_cmk_id** (without **sse_kms**) is specified, the following optional values can be specified with it:
  - **kms_region=***kms_key_region*
  - **access_key_id=***kms_key_user_access_key_id*
  - **secret_access_key=***kms_key_user_secret_access_key*
  - **role=***AWS_IAM_role_name*
- **matdesc**=*json_key_description* - This optional value can be provided only with the *keyinfo* values (**sse_s3**, **sse_kms**, **master_symmetric_key** or **kms_cmk_id**) to specify encryption materials description. If KMS is used (**kms_cmk_id** or **sse_kms**) then **matdesc** must be a JSON object containing only string values.

> Only the combination **sse_kms** with **kms_cmk_id=***aws_kms_key_identifier* is allowed, otherwise only one of the *keyinfo* value must be specified.

For client side encryption, each object is encrypted with an unique AES256 data key. If **master_symmetric_key** is used, this data key is encrypted with AES256, and stored alongside S3 object. If **kms_cmk_id** is used, encryption key is obtained from AWS KMS. By default, HVR uses S3 bucket region and credentials to query KMS. This can be changed by **kms_region**, **access_key_id** and **secret_access_key**. **matdesc**, if provided, will be stored unencrypted alongside S3 object. An encrypted file can be decrypted only with the information stored alongside to the object, combined with master key or AWS KMS credentials; as per Amazon S3 Client Side Encryption specifications. Examples are:

- **master_symmetric_key=123456789ABCDEF123456789ABCDEF123456789ABCDEF123456789ABCDEF**
- **master_symmetric_key=123456789ABCDEF123456789ABCDEF123456789ABCDEF123456789ABCDEF;matdesc={"hvr":"example"}**
- **kms_cmk_id=1234abcd-12ab-34cd-56ef-1234567890ab**
- **kms_cmk_id=1234abcd-12ab-34cd-56ef-1234567890ab; kms_region=us-east-1; access_key_id=AKIAIOFSODNN7EXAMPLE; secret_access_key=wJalrXUtnFEMI/K7DMENG/bPxRfiCYEXAMPLEKEY**
- **kms_cmk_id=1234abcd-12ab-34cd-56ef-1234567890ab; matdesc={"hvr":"example"}**

For server side encryption, each object will be encrypted by the S3 service at rest. If **sse_s3** is specified, HVR will activate SSE-S3 server side encryption. If **sse_kms** is specified, HVR will activate SSE-KMS server side encryption using the default aws/s3 KMS key. If additionally **kms_cmk_id=***aws_kms_key_identifier* is specified, HVR will activate SSE-KMS server side encryption using the specified KMS key id. **matdesc**, if provided, will be stored unencrypted alongside S3 object. Examples are:

- **sse_s3;matdesc={"hvr":"example"}**
- **sse_kms**
- **sse_kms;kms_cmk_id=1234abcd-12ab-34cd-56ef-1234567890ab;matdesc={"hvr":"example"}**

/BulkAPI		Use Salesforce Bulk API (instead of the SOAP interface). This is more efficient for large volumes of data, because less roundtrips are used across the network. A potential disadvantage is that some Salesforce.com licenses limit the number of bulk API operations per day. If this parameter is defined for any table, then it affects all tables captured from that location.
/SerialMode		Force serial mode instead of parallel processing for Bulk API. The **default** is parallel processing, but enabling **/SerialMode** can be used to avoid some problems inside Salesforce.com. If this parameter is defined for any table, then it affects all tables captured from that location.
/CloudLicense		Location runs on cloud node with HVR on-demand licensing. HVR with on-demand licensing can be purchased on-line, for example in Amazon or Azure Marketplace. This form of run-time licensing checking is an alternative to a regular HVR license file (file **hvr.lic** in directory **$HVR_HOME/lib** on the hub), which is purchased directly from HVR-Software Corp. HVR checks licenses at run-time; if there is no regular HVR license on the hub machine which permits the activity then it will attempt to utilize any on-demand licenses for locations defined with this parameter. Note that if HVR's hub is on the on-demand licensed machine then its on-demand license will automatically be utilized, so this parameter is unnecessary.