TableProperties

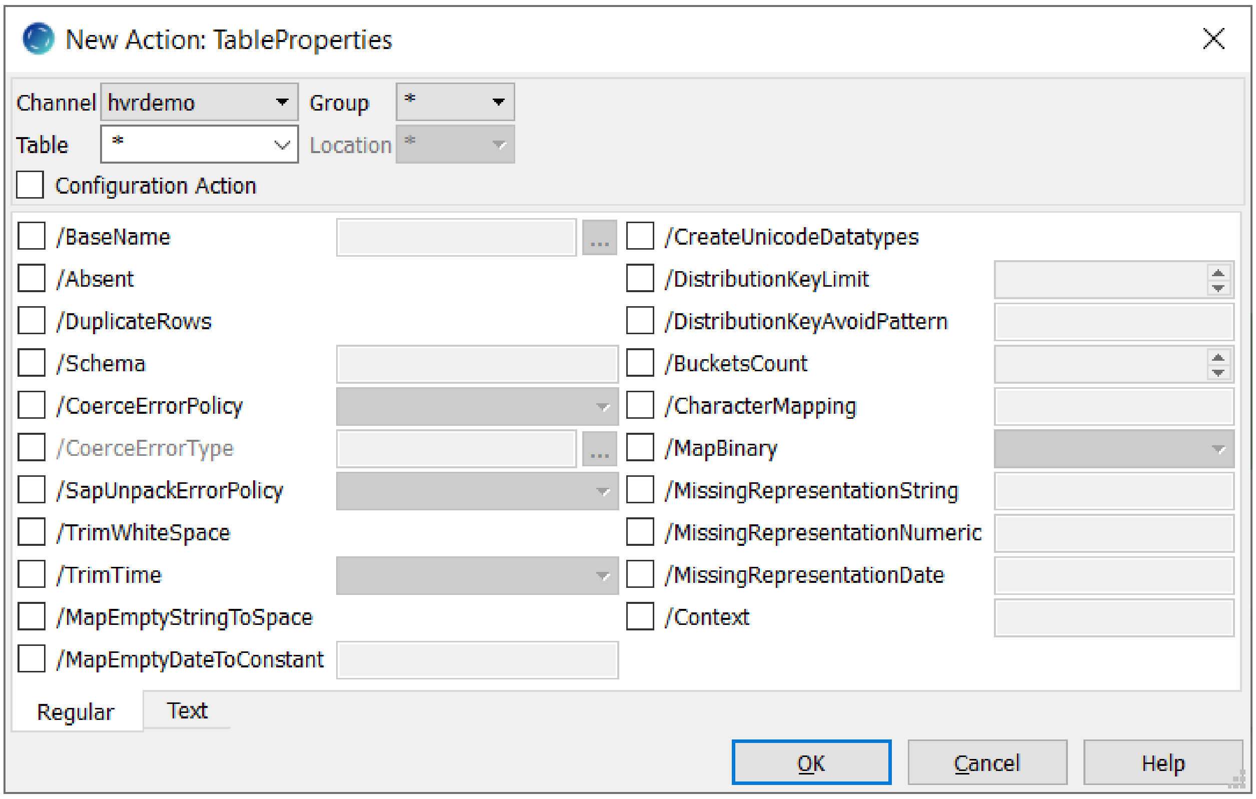
Contents

Description

Action **TableProperties** defines properties of a replicated table in a database location. The action has no effect other than that of its parameters. These parameters affect both replication (on the capture and integrate side) and HVR refresh and compare.

Parameters

This section describes the parameters available for action **TableProperties**.



New Action: TableProperties

Channel hvrdemo
Group *
Table *
Location *
Configuration Action

/BaseName
/Absent
/DuplicateRows
/Schema
/CoerceErrorPolicy
/CoerceErrorType
/SapUnpackErrorPolicy
/TrimWhiteSpace
/TrimTime
/MapEmptyStringToSpace
/MapEmptyDateToConstant
/CreateUnicodeDatatypes
/DistributionKeyLimit
/DistributionKeyAvoidPattern
/BucketsCount
/CharacterMapping
/MapBinary
/MissingRepresentationString
/MissingRepresentationNumeric
/MissingRepresentationDate
/Context

Parameter	Argument	Description

/BaseName	*tbl_name*	This action defines the actual name of the table in the database location, as opposed to the table name that HVR has in the channel. This parameter is needed if the 'base name' of the table is different in the capture and integrate locations. In that case the table name in the HVR channel should have the same name as the 'base name' in the capture database and parameter **/BaseName** should be defined on the integrate side. An alternative is to define the **/BaseName** parameter on the capture database and have the name for the table in the HVR channel the same as the base name in the integrate database. Parameter **/BaseName** is also necessary if different tables have the same table name in a database location but have different owners (**/Schema** parameter). Or if a table's base name is not allowed as an HVR name, e.g. if it contains special characters or if it is too long. If this parameter is not defined then HVR uses the base name column (this is stored in **tbl_base_name** in catalog **hvr_table**). The concept of the 'base name' in a location as opposed to the name in the HVR channel applies to both columns and tables, see **/BaseName** in **ColumnProperties**. Parameter **/BaseName** can also be defined for file locations (to change the name of the table in XML tag) or for Salesforce locations (to match the Salesforce API name).
/Absent		Table does not exist in database. For example, parameter **/Absent** can be defined if a table needs to be excluded from one integrate location but included in another integrate location for a channel with parallel integration jobs.
/DuplicateRows		Replication table can contain duplicate rows. This parameter only has effect if no replication key columns are defined for the table in **hvr_column**. In this case, all updates are treated as key updates and are replicated as a delete and an insert. In addition, each delete is integrated using a special SQL subselect which ensures only a single row is deleted, not multiple rows. When this parameter is used with **Integrate /Burst**, deleting one row from a set of duplicates on the source leads to the deletion of the complete set of duplicate rows on the target. For example, if a table contains five duplicates of a row on the source and if one of them is deleted, this results in deletion of all five rows on the target.
/Schema	*schema*	Name of database schema or user which owns the base table. By default the base table is assumed to be owned by the database username that HVR uses to connect to the database.
/IgnoreCoerceError		No error will be reported if an error is outside the boundary of a destination data type or if a conversion is impossible because of a data type difference (e.g. string '**hello**' must be converted into an integer). Instead HVR will silently round an integer or date up or down so it fits within the boundary, truncate long strings or supply a default value if a value cannot be converted to a target data type. The default value used for date is **0001 01 01**. Since HVR 5.3.1/16, **/IgnoreCoerceError** is replaced with **/CoerceErrorPolicy**.

/CoerceErrorPolicy **Since** v5.3.1/16	*policy*	Defines a *policy* to handle type coercion error (an error which occurs while converting a value from one data type to a target data type). This *policy* typically affects all types of coercion errors, unless parameter **/CoerceErrorType** is defined in the same action. Multiple actions with **/CoerceErrorPolicy** can be defined to apply different policies to different coercion error types. Available options for *policy* are: • **FATAL** (**default**): The replication job fails and displays an error message mentioning the table and column name where the bad values are encountered. • **SILENT** : The bad value is silently (without notification) coerced /replaced with a value that is legal. The coercion details are not written into the job's log file. For details about the how bad values are replaced/coerced, see **/CoerceErrorType** below. • **WARNING** : A warning message is written into the job's log file. The warning message contains the table and column name, the type of coercion (see **/CoerceErrorType** below), and the number of rows affected. • **WARNING_FILE** : The rows with bad values and the values in the key columns are written into a binary file (with extension **.coererr**) on the hub machine and also a warning message is written into the job's log file. This warning contains the table and column name, the type of coercion (see **CoerceErrorType** below), the number of rows affected, and the binary file name. The binary file has the same format as HVR's transaction files. To view its contents, use **Hvrrouterview** command: `hvrrouterview hubdb chn $HVR_CONFIG/router/hubdb/chn/coerceerror/YYYYMMDD/YYYYMMDDHHmmSS-jobname.coererr`

/CoerceErrorType **Since** v5.3.1/16	*types*	This parameter defines which types of coercion errors are affected by **/CoerceErrorPolicy**. The default (if only **/CoerceErrorPolicy** is defined) is for all the below coercion errors to be affected. When multiple *types* are selected, it should be a comma-separated list. Available options for *types* are: • **NUMERIC_RANGE** : When value exceeds the minimum or maximum value allowed in the target numeric data type and the **/CoerceErrorPolicy** is not **FATAL**, the bad value will be replaced with the minimum or maximum legal value. • **DATE_RANGE** : When value exceeds the minimum or maximum value allowed in the target date data type and the **/CoerceErrorPolicy** is not **FATAL**, the bad value will be replaced with the minimum or maximum legal value. • **STRING_TRUNCATION** : When value exceeds the number of bytes or characters allowed in the target string data type and the **/CoerceErrorPolicy** is not **FATAL**, the bad value will be replaced with a truncated value. • **ROUNDING** : When value's precision exceeds the precision allowed in the target date or numeric data type and the **/CoerceErrorPolicy** is not **FATAL**, the bad value will be rounded (rounding toward zero/truncate) to a legal value. • **ENCODING** : When invalid sequences are encountered during encoding from source to target string data type and the **/CoerceErrorPolicy** is not **FATAL**, the the sequence will be sanitized with a replacement sequence. • **NULLS** : When null value is encountered for non nullable target data type and the **/CoerceErrorPolicy** is not **FATAL**, the bad value this will be replaced with default value (a zero or an empty string depending on the data type). • **OTHER** : Anything that does not match any of the above *types* (e.g. non-number string '**hello**' being coerced to a target numeric data type), and the **/CoerceErrorPolicy** is not **FATAL**, this will be replaced with NULL or a default value (zero or empty string) depending on the data type.

/SapUnpackErrorPolicy Since v5.7.5/7	*policy*	Defines a *policy* to handle type coercion error during SapUnpack (when **Transform /SapUnpack** is defined). Type coercion error is an error that occurs while converting a value from one data type to a target data type. This *policy* typically affects all types of coercion errors, unless parameter **/CoerceErrorType** is defined in the same action. Multiple actions with **/CoerceErrorPolicy** can be defined to apply different policies to different coercion error types. Available options for *policy* are: <ul><li>**FATAL** (**default**): The replication job fails and displays an error message mentioning the table and column name where the bad values are encountered.</li><li>**SILENT** : The bad value is silently (without notification) coerced/replaced with a value that is legal. The coercion details are not written into the job's log file. For details about the how bad values are replaced/coerced, see **/CoerceErrorType** below.</li><li>**WARNING** : A warning message is written into the job's log file. The warning message contains the table and column name, the type of coercion (see **/CoerceErrorType** below), and the number of rows affected.</li><li>**WARNING_FILE** : The rows with bad values and the values in the key columns are written into a binary file (with extension **.coererr**) on the hub machine and also a warning message is written into the job's log file. This warning contains the table and column name, the type of coercion (see **CoerceErrorType** below), the number of rows affected, and the binary file name. The binary file has the same format as HVR's transaction files. To view its contents, use **Hvrrouterview** command: `hvrrouterview hubdb chn $HVR_CONFIG/router/hubdb/chn/coerceerror/YYYYMMDD/YYYYMMDDHHmmSS-jobname.coererr`</li></ul>
/TrimWhiteSpace		Remove trailing whitespace from **varchar**.
/TrimTime	*policy*	Trim **time** when converting **date** from Oracle and SQL Server. Available options for *policy* are: <ul><li>**YES** : Always trim **time**</li><li>**NO** (**default**) : Never trim **time**</li><li>**MIDNIGHT** : Only trim if value has **time** as **00:00:00**</li></ul>
/MapEmptyStringToSpace		Convert empty Ingres or SQL Server **varchar** values to an Oracle **varchar2** containing a single space and vice versa.
/MapEmptyDateToConstant	*date*	Convert between Ingres empty date and a special constant *date*. Value *date* must have form *DD*/*MM*/*YYYY*.
/CreateUnicodeDatatypes		On table creation use unicode data types for string columns, e.g. map **varchar** to **nvarchar**

/DistributionKeyLimit	*int*	Maximum number of columns in the implicit distribution key. The **default** value is **1** (just one column). Value **0** means all key columns (or regular columns) can be used. A table's distribution key can be set explicitly or implicitly. An explicit distribution key can be set by clicking the checkboxes in the table's dialog, or by defining parameter **ColumnProperties /DistributionKey**. If no explicit distribution key is defined, then HVR uses the implicit distribution key rule. The implicit rule is to use the first N columns; either from the replication key, or (if the table has no replication key) from the regular columns which do not have a LOB data type. Some DBMS's (such as Redshift) are limited to only one distribution key column.
/DistributionKeyAvoidPattern	*patt*	Avoid putting given columns in the implicit distribution key. For a description of the implicit distribution key, see parameter **/DistributionKeyLimit** above. The **default** value is **''** (no column is avoided). If this parameter is defined then HVR will avoid adding any columns whose name matches to the implicit distribution key. So if the table has replication key columns (*k1 k2 k3 k4*) and **/DistributionKeyAvoidPattern**=*'k2|k3'* and **/DistributionKeyLimit**=*2* then the implicit distribution key would be (*k1 k4*). But if **/DistributionKeyAvoidPattern**=*'k2|k3'* and **/DistributionKeyLimit**=*4* then the implicit distribution key would be (*k1 k2 k3 k4*). For SAP databases, column **'mandt'** is often constant, so parameter **/DistributionKeyAvoidPattern**=*mandt* should be used.
/BucketsCount **Hive ACID**		Number of buckets to be specified while creating a table in Hive ACID. If this parameter is not selected, the **default** value is **1**.
/CharacterMapping	*rules*	Allows to replace some characters (potentially unsupported) in string columns with a replacement sequence. Value *rules* should be a semicolon separated list of elements, each with form *char***>***chars*. Each *char* be a literal character or have form **\n**, **\r**, **\t**, \\, **\x***NN*, **\u***NNNN*, **\U***NNNNNNNN* (where *N* is a hex digit). Example;**"\n>\\n;\r>\\r;\x00>\\0"**. Note that the mapping is performed during integration, refresh and also compare, so the HVR compare does not show an earlier mapping as a difference.
/MapBinary	*policy*	Controls the way binary columns are mapped to a string. This parameter is relevant only if the location does not support any binary data type (e.g. Redshift) (or) **FileFormat /Csv** or **FileFormat /Json** is defined for the location (or) a binary column is explicitly mapped to a string column using **ColumnProperties /DatatypeMatch /Datatype**. Available options for *policy* are: • **COPY** (**default** for CSV and databases): Memory copy of the binary data. This can cause invalid characters in the output. • **HEX** : The binary value is represented as HEX string. • **BASE64** (**default** for Json): The binary value is represented as Base64 string.

/MissingRepresentationString **Since** v5.3.1/5 **File** **Kafka**	*str*	Inserts value *str* into the string data type column(s) if value is missing/empty in the respective column(s) during integration. The value *str* defined here should be a valid input for the column(s) in target database. When **/MissingRepresentationNumeric** or **/MissingRepresentationDate** is used without defining **/MissingRepresentationString** then a default value (for example, empty string) is inserted into the string data type column(s) in which the value is missing/empty. Defining **/MissingRepresentationString** enables HVR to use **ColumnProperties** **/TimeKey** without requiring supplemental logging all.
/MissingRepresentationNumeric **Since** v5.3.1/5 **File** **Kafka**	*str*	Inserts value *str* into the numeric data type column(s) if value is missing/empty in the respective column(s) during integration. The value *str* defined here should be a valid input for the column(s) in target database. When **/MissingRepresentationString** or **/MissingRepresentationDate** is used without defining **/MissingRepresentationNumeric** then a default value (for example, 0) is inserted into the numeric data type column(s) in which the value is missing/empty. Defining **/MissingRepresentationNumeric** enables HVR to use **ColumnProperties** **/TimeKey** without requiring supplemental logging all.
/MissingRepresentationDate **Since** v5.3.1/5 **File** **Kafka**	*str*	Inserts value *str* into the date data type column(s) if value is missing/empty in the respective column(s) during integration. The value *str* defined here should be a valid input for the column(s) in target database. When **/MissingRepresentationNumeric** or **/MissingRepresentationString** is used without defining **/MissingRepresentationDate** then a default value is inserted into the date data type column(s) in which the value is missing/empty. Defining **/MissingRepresentationDate** enables HVR to use **ColumnProperties** **/TimeKey** without requiring supplemental logging all.
/Context **Since** **v5.7.5/6**	*ctx*	Ignore action unless the **Hvrrefresh** or **Hvrcompare** context *ctx* is enabled. The value must be the name of the context (a lowercase identifier). It can also be specified as **!***ctx*, which means that the action is effective unless context *ctx* is enabled. One or more contexts can be enabled for **Hvrcompare** or **Hvrrefresh** (on the command line with option **–C***ctx*). Defining an action that is only effective when the context is enabled can have different uses. For example, if action **TableProperties** with parameters **/BaseName=other_name** and **/Context=diff_base** is defined, then normally the default base name is used, but if context **diff_base** is enabled (**–C diff_base**), then the base name **other_name** is used.