How To...

Master Data Governance for Material: Check and Derivation Rules

Applicable Releases:

From EHP6 FOR SAP ERP 6.0 and from SAP S/4HANA 1511

Version 2.1
May 2017
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
<tr>
<td>1.20</td>
<td>Rule for Backend Check for Net weight</td>
</tr>
<tr>
<td>1.30</td>
<td>Additional notes, General Implementation</td>
</tr>
<tr>
<td>1.40</td>
<td>EhP6: BAdI USMD_RULE_SERVICE_CROSS_ET Chapter 6.1.2</td>
</tr>
<tr>
<td>1.50</td>
<td>New Chapter: Derivation with BAdI USMD_RULE_SERVICE_CROSS_ET</td>
</tr>
<tr>
<td>1.60</td>
<td>New function in MDG6.1 (see chapter 3.4 and Note 1701437)</td>
</tr>
<tr>
<td>1.70</td>
<td>Chapter 4.4, 5.2 and 5.3</td>
</tr>
<tr>
<td>1.80</td>
<td>Small corrections in chapter 5.1 and 5.2</td>
</tr>
<tr>
<td>1.90</td>
<td>New Chapter: Cross Entity Read Access</td>
</tr>
<tr>
<td>2.00</td>
<td>Small corrections</td>
</tr>
<tr>
<td>2.10</td>
<td>Small corrections 4.2.4; new chapters 4.4.4 and 7.7</td>
</tr>
</tbody>
</table>
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example Text</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><code>&lt;Example text&gt;</code></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔴</td>
<td>Caution</td>
</tr>
<tr>
<td>🔵</td>
<td>Note or Important</td>
</tr>
<tr>
<td>🔥</td>
<td>Example</td>
</tr>
<tr>
<td>🔒</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **Business Scenario** ........................................................................................................... 1
2. **Prerequisites** ....................................................................................................................... 1
3. **Definition and Overview** .................................................................................................... 2
   3.1 Definition ........................................................................................................................... 2
   3.2 Overview Checks and Derivations ....................................................................................... 2
4. **Checks** ............................................................................................................................... 3
   4.1 Model driven checks ......................................................................................................... 3
      4.1.1 Code lists .................................................................................................................. 3
      4.1.2 Cardinality check ...................................................................................................... 3
      4.1.3 Format check ............................................................................................................ 3
      4.1.4 Required fields ......................................................................................................... 3
   4.2 Re-use of existing check logic in ERP backend system or S/4HANA backend system ......... 4
      4.2.1 Customizing OMT4 .................................................................................................... 4
      4.2.2 Code list .................................................................................................................. 4
      4.2.3 Business checks (Material API) ................................................................................ 4
      4.2.4 Required fields as defined in material master customizing (T130F) and hardcoded (Material API) .................................................................................................................. 5
   4.3 Custom Checks ................................................................................................................. 6
      4.3.1 Checks using BRF+/BAdI USMD_RULE_SERVICE .................................................. 6
   4.4 Customizing CHANGE REQUEST Type for Check messages .......................................... 6
      4.4.1 Customizing setting: Issue Error as Warnings ......................................................... 7
      4.4.2 Customizing setting: Standard ................................................................................ 7
      4.4.3 Configure Properties of Change Request Step ....................................................... 8
      4.4.4 Enforce check at any roundtrip ............................................................................... 8
5. **Checks with BRF+** ............................................................................................................. 8
   5.1 Example for CHECK_MATERIAL .................................................................................. 9
      5.1.1 Simple Checks ......................................................................................................... 9
   5.2 Example for CHECK_MEAN_GTIN .................................................................................. 10
      5.2.1 Simple Checks ...................................................................................................... 10
   5.3 Example for CHECK_UNITOFMSR ................................................................................ 11
      5.3.1 Simple Checks ...................................................................................................... 11
      5.3.2 Rule to compare Net and Gross weight for Base Unit of Measure ....................... 12
   5.4 Example for CHECK_MARCBASIC ............................................................................... 14
      5.4.1 Warning, if Serial Number Profile is changed ..................................................... 14
6. **Derivations** ....................................................................................................................... 17
   6.1 Custom derivations ......................................................................................................... 17
      6.1.1 Derivation using BRF+/BAdI USMD_RULE_SERVICE .............................................. 17
      6.1.2 EhP6: Derivation using BAdI USMD_RULE_SERVICE_CROSS_ET ...................... 17
      6.1.3 Comparison of Derivations .................................................................................... 17
   6.2 Change Request Enrichment BADI (MDG_BS_MAT_API_ENRICH_BADI) ...................... 18
1. Business Scenario

SAP Master Data Governance for Material (MDG-M) provides business processes to find, create, change, and mark material master data for deletion. It supports the governance of material master data in a central hub and the distribution to connected operational and business intelligence systems.

The processes are workflow-driven and can include several approval and revision phases, and the collaboration of all users participating in the master data maintenance.

To support these processes, check and derivation rules can be created for the MM data model. Checks ensure that the master data is consistent. You can use derivations to calculate values for attributes from other resolved attributes, thereby simplifying data entry.

This How-To Guide gives an overview of the checks used in MDG-M and gives examples for Checks and Derivations built in BRF+.

2. Prerequisites

Check if the following SAP Notes are required and implemented in the system:

- 1592108  Unexpected mandatory fields after model extension
- 199165   Field selection control apparently not working
- 44410    Integrating cust.-specific fields in matl master
- 1696462  BRF+ and BAdI derivations are not performed
- 1731488  After model extension data disappears from UI
- 1764334  Call of derivations only at end of write loop
- 2100135  Incorrect required field check for execution via BAdI

- BRF plus documentation
  - Useful transaction FDT_HELPERS
  - Online help [http://help.sap.com/erp2005_ehp_06/helpdata/EN/9a/6b67ce7c26446483af079719edf679/content.htm?frameset=/EN/00/da13e29b4e4fb4ace5990f212a05da/frameset.htm](http://help.sap.com/erp2005_ehp_06/helpdata/EN/9a/6b67ce7c26446483af079719edf679/content.htm?frameset=/EN/00/da13e29b4e4fb4ace5990f212a05da/frameset.htm)
  - BRF plus Application Exits [http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/d0852307-7ad7-2d10-dbb7-cb1f0a21496e](http://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/d0852307-7ad7-2d10-dbb7-cb1f0a21496e)
3. Definition and Overview

3.1 Definition

- Checks raise messages. Message with severity *info*, *warning*, *error* and *abort* are possible.
- Derivations calculate values for attributes and can also raise messages. Only messages with severity *info* are possible.

3.2 Overview Checks and Derivations

- Model driven checks
  - Code list check
  - Cardinality check
  - Format check
  - Required fields

- Re-use of existing check logic in ERP backend system or S/4HANA backend system
  - Check messages configured in OMT4
  - Code lists
  - Business checks (Material API)
  - Required fields as defined in material master customizing (T130F) and hardcoded (Material API)

- Custom checks (modeled and programmed)
  - Rules from the Business Rules Framework (BRF+)
  - Coded rules using the BAdI USMD_RULE_SERVICE

- Custom derivations (modeled and programmed)
  - Rules from the Business Rules Framework (BRF+)
  - Coded rules using BAdI USMD_RULE_SERVICE
  - Coded rules using BAdI USMD_RULE_SERVICE_CROSS_ET
  - Coded enrichments using the BAdI MDG_BS_MAT_API_ENRICH_BADI
  - Derivation with SMT-Mapping

Checks are usually carried out at *Check*, *Run Validation*, *Save*, and *Activate* while derivations are carried out each roundtrip.

In addition, input errors found by basic technical checks (like for Units of Measure, Dates, and Currencies) are always executed after the next round trip and issued as errors.
4. Checks

4.1 Model driven checks

4.1.1 Code lists

The considered Code list for the check comes from the Fixed Values or Value Range Table which is assigned to the domain of the data element.

In the customizing for the MM data model you can use the ‘No Existence Check’ to deactivate the standard existence check for the value of the attribute.

Path: Master Data Governance -> General Settings -> Data Modeling -> Edit Data Model (VC_USMD001)

4.1.2 Cardinality check

A field is mandatory if the referencing relationship has cardinality 1:1.

4.1.3 Format check

Data element is used for format check.

4.1.4 Required fields

Attributes can be defined as required fields.
4.2 Re-use of existing check logic in ERP backend system or S/4HANA backend system

4.2.1 Customizing OMT4

Only messages with severity E from the material API are shown in MDG. These can be messages for the required fields or additional messages defined in the customizing OMT4. (See Customizing, under Logistics -> General -> Material Master -> Basic Settings -> Define Attributes of System Messages).

With the customizing of the change request type you can decide if these E messages are shown as errors or as warning messages during the governance processes. But at activation there is no conversion of the messages.

New function in MDG6.1 (see also Note 1701437, Chapter Message configuration in OMT4): Transaction OMT4 provides configuration for message severity. Messages can be raised as errors, warnings, or not at all. MDG-M only supports the severity configuration for a subset of these messages.

- If a message from this subset is configured as a warning, it is also shown as warning in MDG-M
- If a message not in this subset is configured as a warning, it is not shown in MDG-M
- Messages configured as errors are always shown as errors.

Supported subset:

- M3, configuration for messages 132, 159, 285, 347, and 348 is supported
- MM, configuration for messages 189, 312, and 657 is supported
- MH and WE, configuration for messages is not supported

4.2.2 Code list

The Material API checks the Fixed Values or Value Range Tables which are assigned to the domain of the data elements in the material master tables.

4.2.3 Business checks (Material API)

The Material API is called at Check, Run Validation, Save, and Activate

- Check and Save only check the changed material
- *Run Validation* checks the complete change request
- In Single Item Maintenance UI and Mass Maintenance UI, File import, etc.

Some of the settings of the Material Master Customizing are considered in MDG through the Material API.

All settings from the section Basic Settings are considered. For example, the output format of the material number and the material types. All settings from the section Settings for Key Fields are considered. For example, the definition of material groups and basic material and the settings for EAN’s.

In addition, User Exit (SMOD) MGA00001 and BADI_MATERIAL_CHECK are considered in MDG-M.

### 4.2.4 Required fields as defined in material master customizing (T130F) and hardcoded (Material API)

The field properties for required fields in the Material Master Customizing are considered in the MDG checks.

The section Field Selection is considered. Changes affect the field’s properties in MDG-M. There, you define if a field is required (table T130F).

The field properties are determined by the field selection group. For the specification of the properties of a field selection group, there are several criteria. These are called field selection references. Six criteria (field selection references) are relevant for the definition of the field selection attributes (see consulting note 199165).

The six field selections which are relevant for MDG_M are: Industry sector, Material type, transaction MM01/MM02 and field selection control KB. Also considered are SAP1 and SAP2, but must not be changed by customer. As MDG can’t distinguish between the backend material create and change transaction both MM01 and MM02 checks are considered in the Check.

For the determination of UI field property values, see chapter 5.5. in [http://scn.sap.com/docs/DOC-30192](http://scn.sap.com/docs/DOC-30192). Here only the settings for MM01 are considered for performance reasons, not MM02. Therefore, SAP recommends keeping the field control for MM01/MM02 in sync to avoid nontransparent messages.

Also, BADI_MAT_F_SPEC_SEL is considered in MDG-M.
4.3 Custom Checks

4.3.1 Checks using BRF+/BAdI USMD_RULE_SERVICE

Check Entities from the BRF+ Rules and the Check BAdI are called at Check, Run Validation, Save, and Activate.

Check Change Request from the Check BAdI and the BRF+ Rules is called at Run Validation and Activate.

- Check/Save only check the changed Material
- Run Validation checks the complete Change Request
- Valid for Single Item Maintenance UI and Mass Maintenance UI, File upload, Data import …

4.4 Customizing CHANGE REQUEST Type for Check messages

Only messages with type error from the backend Material API are shown in MDG-M. Additional messages for checks can be created with BRF+ or with a BADI. These can be shown as abort, errors, warnings, or info. Within the customizing of the change request type you can decide if these are shown as errors or as warning messages.

Note: Input errors found by basic technical checks (like for Units of Measure, Dates, and Currencies) are always issued as Errors.
4.4.1 Customizing setting: Issue Error as Warnings

<table>
<thead>
<tr>
<th>EhP5 and * Issue Error as Warnings</th>
<th>MAT01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round trip</td>
<td>Check</td>
</tr>
<tr>
<td>Derive Entity BRF+</td>
<td>A→E;</td>
</tr>
<tr>
<td></td>
<td>E→W;</td>
</tr>
<tr>
<td>Derive BADI</td>
<td>A→E;</td>
</tr>
<tr>
<td></td>
<td>E→W;</td>
</tr>
<tr>
<td>Check Entity BRF+ *</td>
<td>n.a.</td>
</tr>
<tr>
<td>Check Entity BADI *</td>
<td>n.a.</td>
</tr>
<tr>
<td>Check Material API (only A, E messages) *</td>
<td>n.a.</td>
</tr>
<tr>
<td>CR check’s (only E messages)</td>
<td>n.a.</td>
</tr>
<tr>
<td>Check CR BRF+</td>
<td>n.a.</td>
</tr>
<tr>
<td>Check CR BADI</td>
<td>n.a.</td>
</tr>
<tr>
<td>Duplicate Check</td>
<td>n.a.</td>
</tr>
</tbody>
</table>

Messages from Derives are allows converted into info messages.

4.4.2 Customizing setting: Standard

<table>
<thead>
<tr>
<th>EhP5 and Standard ZZMAT01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Round trip</td>
</tr>
<tr>
<td>Derive BRF+</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Derive BADI</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Check Entity BRF+ *</td>
</tr>
<tr>
<td>Check Entity BADI *</td>
</tr>
<tr>
<td>Check Material API (only A, E messages) *</td>
</tr>
<tr>
<td>CR check’s (only E messages)</td>
</tr>
<tr>
<td>Check CR BRF+</td>
</tr>
<tr>
<td>Check CR BADI</td>
</tr>
<tr>
<td>Duplicate Check</td>
</tr>
</tbody>
</table>

Messages from Derives are allowd converted into info messages.
4.4.3 Configure Properties of Change Request Step

In this Customizing activity, you determine settings for the execution of a change request step for a change request type.

Enrichment Spots and Checks per Change Request Step View

In this view, you can complete the following actions for a change request step:

- Specify which enrichment spots and checks are relevant.
- View the execution sequence for enrichment spots and checks.
- Control the display of messages by specifying a message output. For example, you can ensure some messages display only as warnings.
- Determine whether a check is always executed or only executed when changes occur.

Entity Types per Change Request Step View and Attributes per Change Request Step View

You can complete the following actions for a change request step:

- Specify which fields are relevant, and which relevant fields are required, by setting field properties. For example, you can make a required field optional.
- Reduce the number of checks applied to fields by specifying a Check Logic for an entity type.
- You can only define properties for entity types, attributes, or relationships that are governed. You can apply this setting in Customizing for Master Data Governance under General Settings -> Process Modeling -> Define Governance Scope.

For the field properties topic please see also the How To Guide for the UI. You can find it here: http://scn.sap.com/docs/DOC-30192. See chapter for Field Properties.

4.4.4 Enforce check at any roundtrip

You can enforce a check by any roundtrip in the material UI by enhancing the method CHECK_DATA of the class CL_MDG_BS_BOL_TRANSACTION.

You can enhance with an overwrite exit the IF statement checking the event ID or you can implement a post exit enhancement, copying the core code of the IF statement. In any case, you should care for notes or enhancements of this method in future to transfer the corrections to your enhancement in an appropriate manner.

The event ID ‘XROUNDTRIP’ is raised, if the user presses the Enter key, and if a value help is closed (to refresh the description of the value). If necessary, you can also check for the event ID ‘ROUNDTRIP’ additionally. Depending on your intension, you can check only for one of these event IDs or for both. Check in independent scenarios, which combinations provides the right number of checks in your case.

5. Checks with BRF+

Path: MDGIMG -&gt; General Settings -&gt; Data Quality and Search -&gt; Define Validation and Derivation Rules.

In this Customizing activity, you define the validations and derivations for a data model. This activity calls the Web Dynpro application Definition of Rules for Validations and Derivations (USMD_RULE). You can define validation and derivation rules for the data model MM. The system automatically generates the data objects used for the rule for the selected data model. For a precise description of the procedure, choose Help on the initial screen of this Web Dynpro application.
Important

There is a Naming Convention for Trigger function nodes in the catalog structure. The naming convention for check trigger function nodes of a catalog structure is CHECK_<name of entity type>, for example, CHECK_MATERIAL. The naming convention for derivation trigger function nodes of a catalog structure is DERIVE_<name of entity type>, for example, DERIVE_MATERIAL.

Important

If you update/change the data model MM, the data objects are updated. But if you used a data object already in a function as a signature, it is not automatically updated. Please see SAP Note 1731488 for how to proceed.

5.1 Example for CHECK_MATERIAL

5.1.1 Simple Checks

(5) Rule: No description is available - Unlimited Validity

If

- MATERIAL-Metl Type is equal to HALB
- MATERIAL-Metl Type is equal to NLAG

Then

Perform the following operations

(1) Perform Action FERT_MESSAGE
You can also use Message with message class USMD5 a message number 000, where you can specify variables.

5.2 Example for CHECK_MEAN_GTIN

5.2.1 Simple Checks

(1) Rule: No description is available - Unlimited Validity

If

Europeis...-EAN.UPC is initial

Then

Perform the following operations

(1) Perform Action MESSAGE.EAN
5.3 Example for CHECK_UNITOFMSR

5.3.1 Simple Checks

(2) Rule: No description is available - Unlimited Validity

If

is initial
Menge...Height

or is equal to 0

or

is initial
Menge...Length

or is equal to 0

or

is initial
Menge...Width

or is equal to 0

Then

Perform the following operations

(1) Perform Action MESSAGE_EAN

(2) Perform Action MESSAGE_BREITE
5.3.2 Rule to compare Net and Gross weight for Base Unit of Measure

There is a check in the backend transactions for Material which compares the Net and Gross weight for the Base Unit of Measure. If the net weight is greater than the gross weight, the message ‘The net weight is greater than the gross weight’ is raised (see TA SE91 Message class M3 and message 176).

The backend check does not show up in MDG, because message M3 176 is not in the supported subset (see chapter 4.2.1 Customizing OMT4). Therefore, this message is only shown in MDG if the severity is an error message.

Solution:
You can create a BRF+ rule if the check is also necessary in MDG.
5.4 Example for CHECK_MARCBASIC

5.4.1 Warning, if Serial Number Profile is changed

Scenario:
There should be a warning message when you change the Serial Number Profile (SERNP) for a material which has stock.

Solution:
Create a ruleset for function CHECK_MARCBASIC.
The first rule should check if stock exists for the material after checking that the serial number profile was changed.
The second rule should raise a message if first rule is true.

Step by Step:
- Create ruleset
  - Assign variable BOOLEAN
How To... Check and Derivation Rules for MDG Material

- Create database lookup CHECK_MARC

- Create database lookup CHECK_STOCK

- Create Log Message
• Create Rules
6. Derivations

6.1 Custom derivations

When creating a derivation, ensure that it is only executed if needed because the data is marked as changed and if you close the UI before saving it a confirmation popup appears. See also customer message 101812012: “Unsaved data confirmation popup after request submission”.

6.1.1 Derivation using BRF+/ BAdI USMD_RULE_SERVICE

Derive Entity from BRF+ Rules and Derivation BADI are called at each Entity (per roundtrip).

- Valid for Single Item Maintenance and UI Multi-Record Processing, File upload, Data import etc.
- Not valid for application USMD_MASS_CHANGE. There you select a set of entities of one type, and change single attributes for all these entities to the same value. The derivation in that application is not supported, as it would change other attributes than the specified ones and would break the logic of that application, especially the visualization of the mass change result.
- Messages from Derives are always converted into info messages.

6.1.2 EhP6: Derivation using BAdI USMD_RULE_SERVICE_CROSS_ET

The BAdI is called at least once at every roundtrip (Technical note: the BAdI is triggered in the Flush Method of the Governance API).

- Valid for Single Item Maintenance UI, Multi-Record Processing, application USMD_MASS_CHANGE and Data import
- Messages from Derives are always converted into info messages.
- The BAdI gets only the attributes that are changed since the last BAdI call. Unchanged attributes are not transferred by the interface of this BAdI. You need to use the IO_MODEL parameter (method READ_CHAR_VALUE or RETRIEVE_CHAR_VALUE) to determine their values. For more information, see the documentation of the interface IF_USMD_MODEL_EXT.
  
  Hint: Set parameter IF_NO_FLUSH to ABAP_TRUE to avoid additional call of the Flush Method.

For more Information see chapter 9 Cross-Entity Derivation with BAdI USMD_RULE_SERVICE_CROSS_ET

6.1.3 Comparison of Derivations

It is possible to use all 3 types of Derivations in parallel. Recommendation for the BAdI’s is to use USMD_RULE_SERVICE for derivation if possible for reasons of transparency (self-contained change of entity) and USMD_RULE_SERVICE_CROSS_ET only if this is necessary. The table below gives suggestions for the use of the different custom derivations:

<table>
<thead>
<tr>
<th></th>
<th>BRF+</th>
<th>BAdI USMD_RULE_SERVICE</th>
<th>BAdI USMD_RULE_SERVICE_CROSS_ET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create new entity</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Change one existing entity</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
**6.2 Change Request Enrichment BADI (MDG_BS_MAT_API_ENRICH_BADI)**

You can use this BAdI to automatically fill in fields from the material master that are not subject to master data governance.

- The changes to the data have no impact on the UI
- This BAdI will only be called during activation, when the data is being transferred back to the active area. Therefore, use only technical error messages, no business checks.

**6.3 SMT-Mapping Staging → Active Area**

You can use SMT mapping to automatically fill in fields from the material master that are not subject to master data governance.

- The changes to the data have no impact on the UI
- These defaults are considered during activation and the various checks
- No messages possible
- Use case: To fill in mandatory fields those are not part of the MM data model. For example, to automatically fill in the cross-plant material status with a fixed value (EhP6)
7. Derivations with BRF+

Note:
Derivation for Unit of Measures: If you want to derive Unit of Measures you should use the BAdI USMD_RULE_SERVICE instead of BRF+, as there are technical limitations by using BRF+ for referenced fields.

7.1 Cross-Entity Rules

MDG supports certain cross-entity validations and derivations. Use cases for this include where checks and derivations span across 2 entities, one being a leading entity of storage and use type 1 and one dependent entity of storage and use type 4.

Example for entity MEAN_GTIN with a 1: n relation:
Entities MATERIAL and MEAN_GTIN for the derivation DERIVE_MEAN_GTIN.

Add the MATERIAL to the signature in addition to MEAN_GTIN as you seen on the following picture.

The derivation is triggered during a roundtrip when a WRITE_ENTITY is called for this entity.

Note:
In the entity MATERIAL, of storage and use type 1, you also find the attribute of the 1:1 related dependent entities of type 4 (such as MARASALES and MARAPURCH), but these dependent entities cannot be used for reading or writing. If you want to have derivations for the attributes of these type 4 entities, you should create a derivation like for the 1: n related entities.

Example for entity MARASALES with 1:1 relation:
Entities MATERIAL and MARASALES for the derivation DERIVE_MARASALES.
The derivation is triggered during a roundtrip when a WRITE_ENTITY is called for this entity.

With EhP6 we support cross entity validation and derivation with the BAdI definition USMD_RULE_SERVICE_CROSS_ET. It is also possible to read the classification data and update other entity values with this BAdI (see chapter 9 Cross-Entity Derivation with BAdI USMD_RULE_SERVICE_CROSS_ET). With MDG7.0 it is possible to update the data of an entity (called a target entity) based on the data of another entity (called a source entity) (see chapter 8 Cross-Entity Read Access).

### 7.2 Defaulting of values

With SAP Note 1876796 (prerequisite SAP Note: 1879515) it is possible to default values during material creation/change. BRF+ application is called during start of the OVP for 'Create/Change Material'. As no user input was possible until this point in time, the rule for defaulting cannot have any conditions. Expressions like if field A has value 'X', then default field B with 'Y' fail because the staging area is empty for this material. Only data like, for example, user or change request type can be used in these conditions.

Be aware that these rules are also executed during the normal maintenance. If the user should be able to overwrite the default values; you should check for initial fields first.

Defaulting will only be executed if entity doesn’t exist yet and all conditions of rules are valid. That means defaulting is executed at this point in time when entity is created.
Example: Defaulting for the transportation group on MARASALES

7.3 Example for DERIVE_MATERIAL

7.3.1 Execution time

Derive Entity from BRF+ Rules and Derivation BADI are called at each Entity (per roundtrip).
Make sure that the relevant field is maintained accordingly in your UI Configuration.
### 7.3.2 Simple Rules

1. **Rule:** No description is available - Unlimited Validity

   **If**
   
   True is equal to true

   **Then**
   
   Perform the following operations:
   
   (1) Change value of MATERIAL industry to M

### 7.3.3 Example: Rules with Expression

1. **Rule:** No description is available - Unlimited Validity

   **If**
   
   True is equal to true

   **Then**
   
   Perform the following operations:
   
   (1) Change MATERIAL-Prod.hier after processing expression PRDHA_FORMULA
How To... Check and Derivation Rules for MDG Material
How To... Check and Derivation Rules for MDG Material

(4) Rule: No description is available - Unlimited Validity

If

True is equal to true

Then

Perform the following operations

(1) Change MATERIAL Base type after processing expression FORMULA BASE TYPE
How To... Check and Derivation Rules for MDG Material

7.3.4 Example: Decision Table with multiple results

You want to return multiple results from one decision table. This requires that you pass back the results to the rule using a structure data object.

Here you find an example with BRF+ Decision Table for derivation with 2 output fields.

7.3.4.1 Create a Structure Data Object

http://help.sap.com/erp2005_ehp_05/helpdata/EN/00/da13e29b4e4fb4ace5990f212a05da/frameset.htm
How To... Check and Derivation Rules for MDG Material

Add Elements to the created Structure. Under the Detail section, choose Add Existing Data Object.
Choose *Activate* and save the data object.

### 7.3.4.2 Create a Decision Table

[http://help.sap.com/erp2005_ehp_05/helpdata/EN/00/da13e29b4e4f4ace5990f212a05da/frameset.htm](http://help.sap.com/erp2005_ehp_05/helpdata/EN/00/da13e29b4e4f4ace5990f212a05da/frameset.htm)

Add Result to the Decision Table

Choose [Graphical Access](#) next to Result Data Object.
How To... Check and Derivation Rules for MDG Material

Table Settings

Object(s) created successfully

Result Data Object
- [ ] Return all matches found
- [ ] Return an initial value if no match is found
- [ ] Return an exception if partial match is found
- Result Data Object: [ ]

List of Columns

<table>
<thead>
<tr>
<th>Condition Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Column</td>
</tr>
<tr>
<td>Column name</td>
</tr>
<tr>
<td>No condition columns have been created yet.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Result Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Column from Data Object</td>
</tr>
<tr>
<td>Column name</td>
</tr>
<tr>
<td>No result columns have been created yet.</td>
</tr>
</tbody>
</table>

Table Check Settings

Table Conformance Check Settings: Application Default
Table Gap Check Settings: Application Default

[OK] [Cancel]
How To... Check and Derivation Rules for MDG Material

Add Context to the Decision Table

[Diagram of a decision table with options for inserting columns from context data objects and expressions]
7.3.4.3 Create a Rule

http://help.sap.com/erp2005_ehp_05/helpdata/EN/00/da13e29b4e4fb4ace5990f212a05da/frameset.htm
How To... Check and Derivation Rules for MDG Material
7.3.4.4  Create a Ruleset

http://help.sap.com/erp2005_ehp_05/helpdata/EN/00/da13e29b4e4f4ace5990f212a05da/frameset.htm
Now the system returns the multiple results Size/dim. and Format from the decision table depending on the maintained material type.

### 7.4 Example for DERIVE_MARCMRPFC

#### 7.4.1 Default the Period Indicator to ‘M’

**Scenario:**
The Period Indicator (MARC-PERKZ) should be defaulted as in the backend (FM MARC_ADD_REFERENCE_AFT).
Solution:
Create a ruleset for function DERIVE_MARCMRPFC.
Create a rule for if period indicator (MARCMRPFC-PERMRPFC) is initial set it to ‘M’.

7.5 Example for DERIVE_MARCMRPSP

7.5.1 Default the Procurement Type

Scenario:
The Procurement Type (MARC-BESKZ) should be defaulted as in the backend.

<table>
<thead>
<tr>
<th>T134-BSEXT</th>
<th>T134-BSINT</th>
<th>MARC-BESKZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>blank</td>
</tr>
<tr>
<td>0</td>
<td>1 or 2</td>
<td>E</td>
</tr>
<tr>
<td>1 or 2</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td>1 or 2</td>
<td>1 or 2</td>
<td>X</td>
</tr>
</tbody>
</table>

Solution:
Create a ruleset for function DERIVE_MARCMRPSP.
Create rules for exit conditions and create the following rule for defaulting: if material type is not initial then change procurement type according to decision table.

Step by Step:
- Create ruleset
- Create variable for structure with BSEXT, BSINT, and MTART and expression to initialize the variable.
How To... Check and Derivation Rules for MDG Material

Structure: IS_T134

General

Define Data Binding
Binding Type: No Binding

Structure Definition
- Add Existing Data Object
- Add New Data Object

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Text</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSEXT</td>
<td></td>
<td>External POs</td>
</tr>
<tr>
<td>BSINT</td>
<td></td>
<td>Internal POs</td>
</tr>
<tr>
<td>MTART</td>
<td></td>
<td>Material type</td>
</tr>
</tbody>
</table>

Database Lookup: READ_T134

General

Detail

Selection Mode: Data Retrieval
- View SQL
- Context Overview
- Start Simulation

Select: Single Entry
- T134
- Material Types

With Condition: MTART = MATERIAL-MTART
- Change

Field Mapping

<table>
<thead>
<tr>
<th>Source Field</th>
<th>Text</th>
<th>Target Field</th>
<th>Text</th>
<th>Aggregation</th>
<th>Group By</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSEXT</td>
<td>External Purchase Orders Allowed</td>
<td>BSEXT</td>
<td>External POs</td>
<td>Select</td>
<td></td>
</tr>
<tr>
<td>BSINT</td>
<td>Internal purchase orders allowed</td>
<td>BSINT</td>
<td>Internal POs</td>
<td>Select</td>
<td></td>
</tr>
<tr>
<td>MTART</td>
<td>Material type</td>
<td>MTART</td>
<td>Material type</td>
<td>Select</td>
<td></td>
</tr>
</tbody>
</table>

Ruleset: RS_MARCMPSP, RS_MARCMPSP

General

Detail

Hide-Ruleset Header: Context Overview

Enabled: Yes

Precondition: DERIVE_MARCMPSP

Variables

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#IS_T134ap</td>
<td></td>
</tr>
<tr>
<td>#&lt;id asgn&gt;</td>
<td></td>
</tr>
</tbody>
</table>

Expressions Which Initialize the Variables

1. Update variables after processing READ_T134
2. Not assigned
Create rules for exit conditions (material type is initial; procurement type not initial in staging; procurement type in MARC active area is 'blank') and a database lookup to check if MARC has the value 'blank' for the procurement type.

**Rules**
- **Disable Condition (1) Exit Condition: Material type is initial - Unlimited Validity**
  - Exit rule set when `MATERIAL-MTART` is initial
- **Disable Condition (2) Exit Condition: Procurement is not initial - Unlimited Validity**
  - Exit rule set when `MARC-MRPS-P-BSKZ` is not initial
- **Disable Condition (3) Exit Condition: Read MARC-BSKZ - Unlimited Validity**
  - Exit rule set when `MARC-SELECT` is true

**Database Lookup: MARC_SELECT, Read MARC-BSKZ**
- If at least one entry exists in table MARC
- With condition: `BSKZ` is initial and `MATERIAL` is equal to `MARC-MRPS-P-MATERIAL` and `WERKS` is equal to `MARC-MRPS-P-WERKS`
- Then `BOOLEAN` is True, Otherwise It is False.
• Create rule for defaulting with decision table

Example:
Using variables in Derivation Rules (like Workflow step)

You want to derive values or messages dependent on a workflow step. The workflow step is not in the context of the BRF+ (until MDG6.1). But you can use variables, and you need to get the values. In the UI BAdI (EhP5) or enhanced METHOD process_event (EhP6) the value is set and in the BRF rule the value is retrieved.

7.6.1 Set the values for the variables

7.6.1.1 EhP5: Set the values (EhP5 UI BAdI)

In the process, you should choose a Change Request Type that uses the UI configuration that is the filter value in the BAdI.
How To... Check and Derivation Rules for MDG Material

Class Builder: Class ZEM_MAT_UI_BADI Display

Method DF_EX_UUIU_EVENT2-INITIALIZE

```java
1  METHOD if_es_uuiu_event2-initialize.
   2    CALL METHOD super-if_es_uuiu_event2-initialize.
   3       EXPORTING
   4           is_ui_context = is_ui_context.
   5           is_key_field = is_key_field.
   6       IMPORTING
   7           ot_id_field = ot_id_field.
   8           ot_message = ot_message.
   9       CHANGED
   10          ot_data = ot_data.
```

Class Builder: Class ZL_IBTEST_WF_TOOLS Display

Method GET_WF_STEP_UI_BADI

```java
3  METHOD GET_WF_STEP_UI_BADI.
4      " Doesn't do much, does it?"
5      ot_step = iv_wf_step.
6      ENDMETHOD.
```
7.6.1.2 EhP6: Set the values for the variables

Read the WF container using CL_USMD_APP_CONTEXT and use the step_action to fill your variable into a BADI.

Enhance METHOD process_event (CL_MDG_BS_CR UIBB_EVENTHANDLER). With IO_EVENT->MV_EVENT_ID you will get the action and you can fill your variable.

7.6.1.3 MDG6.1: New variables

Now the workflow step is added in the context of the BRF+.
7.6.2 Create BRF+ Ruleset

The image shows a screenshot of the SAP cockpit interface for creating a BRF+ ruleset. The ruleset is named `Derive_material_02` and is configured to check if `DERIVE_MATERIAL` is processed. The ruleset contains one rule that is triggered if `DERIVE_MATERIAL` is processed. The ruleset variables are initialized by processing the following expressions:

- Expression 1: Update variables after processing expression `IF STEP = 1, THEN`
7.6.3 Create BRF+ Function
7.6.4 EhP5 screenshots of the process

Create Material with CHANGE REQUEST-Type EM_MAT05.
7.7 Example: BRF Validations/Derivations for staging only

7.7.1 Background

Checks and derivations are called during change request processing and records are checked against these rules. In MDGIMG you can define on change request type and step level, if the checks are executed and if the checks should be handled as errors or warnings only.

The rules are executed for all entities and organization units; independently which organization unit is in the staging of the corresponding change request.

Example: a material is in active area and opened for 2 plants (plant 1000 and 2000). A change request is started and only 1 plant (1000) is modified; hence only plant 1000 is in staging. If a user clicks on CHECK in the MDG UI, the system will execute the check for plant 1000 and 2000. This might lead into business process issues, if the user executing the CHECK is only responsible for plant 1000 and has no authorization to change/correct plant 2000.

This chapter explains a solution to be able to execute the BRF+ Checks only for the plant which is in the staging area.

7.7.2 Approach

The approach is to provide a solution without a modification. The solution will add an additional BRF+ rule or precondition to the BRF+ ruleset on the specific entity. The additional rule executes a BRF+ procedure call. This procedure call calls an ABAP class in the backend to retrieve from an ABAP coding the plant which is currently in the staging area of the corresponding change request.
How To... Check and Derivation Rules for MDG Material

Afterwards the value will be saved into the context of the BRF function and can be used for filters in every other rule.

### 7.7.3 Step by Step

1. Implement ABAP class
   
   a. Implement a ABAP class in SE80 which implements the interface IF_FDT_APPLICATION_SETTINGS like a class ZMDGM_PLANT.

   ![Class/Interface Table](image1.png)

   b. Create a new method on class ZMDGM_PLANT like “GET_STAGING_PLANT”. The method needs at least to parameter:
      
      i. Importing: CHANGE REQUEST ID
      
      ii. Exporting: single plant

   ![Parameters of Method](image2.png)

   c. Implement the details of the method. For this you can use MDG API and the method read_char_value. In this method, you should use the readmode if_usmd_model_ext=>gc_readmode_no_act which means that only staging area will be read.

2. BRF+

   a. Create Data object „STAGING PLANT” of Type WERKS_D and activate.
b. Navigate to the Check (like CHECK: MARCBASIC) and to the corresponding ruleset.

c. Add the Change Request ID to the context/signature.

d. Add a new rule (position 1) to fill the context element STAGING_AREA by calling the ABAP class you created earlier.

e. Use in Rule (or Precondition) to check.
How To... Check and Derivation Rules for MDG Material

**Ruleset: CHECK_MARC BASIC**

**General**

**Detail**

- Show Ruleset Header
- Context Overview

**Rules**

- Insert Rule
- Insert Exit Condition

1. **Disable Rule**
   - (1) Rule: No description is available - Unlimited Validity
   - **(1) Change** [STAGING_PLANT after processing] [ZMDOM_CALLCLASS_2]

2. **Disable Rule**
   - (2) Rule: No description is available - Unlimited Validity
   - **If** [MARC BASIC-Plant is equal to STAGING_PLANT]
   - **Then** (1) Perform [ZSULTEST]
8. Cross-Entity Read Access for Derivation BRF+

With MDG7.0 it is possible to update the data of an entity (called a target entity) based on the data of another entity (called a source entity).

In addition, it is possible, by implementing a post-exit, to enforce the BRF+ derivation calls for a target entity, if only the data of the source entity is changed.

Note, that the BRF+ entity derivation is only called if data already exists for the target entity.

8.1 Example: Change the MRP Type Depending on the Plant-Specific Material Status

Scenario:
You want to create a BRF+ derivation to update the field MRP Type (DISMM) and field MRP Group (DGRMRPPP) of the target entity MARCMRPPP depending on the value of the field Plant-Specific Material Status (MMSTA) of the source entity MARCBASIC according to the following logic:

\[
\text{IF } \text{value(MMSTA)} = '40' \text{ and value(DISMM)} = 'ND' \text{ and value(DGRMRPPP)} = 'initial' \\
\text{THEN } \text{set value(DISMM)} = 'PD' \text{ and set value(DGRMRPPP)} = '0001' \\
\text{ENDIF}
\]

Solution:
On the UI, the first field (MMSTA) appears in the assignment block Plant: General Data and the target fields (DISMM and DGRMRPPP) appear in the assignment block Plant: Material Requirements Planning.

Realization:
1: Create a BRF+ derivation with a ruleset containing the following rules:
   - A rule to generate a table variable containing the key values of the source entity
   - A rule containing a procedure call having as input parameter the table generated in the previous rule

2: Enforce the BRF+ derivation if only data of the source entity will be changed and there is data for the target entity through post-exit.

8.1.1 Prerequisite: Create Data Object

First following prerequisite needs to be done only once for data model MM.

Open the transaction USMD_RULE for data model MM. Create a data object (call it ATTRIBUTE_VALUE_01) of type Element to be able to save an attribute value. Right click on the
node DATA_OBJECT and navigate to the entry Element as shown in the next screen:

On the screen enter a Name of your element, for example ATTRIBUTE_VALUE_01, a Short Text and a Text. For the data binding select the entry Bind to DDIC element (Data Dictionary). On the new screen enter the value usmd_value in the field DDIC Element. Choose OK.
Choose *Create* and accept your decisions on the screens.

### 8.1.2 Prerequisite: Create Post-Exit

First following prerequisite needs to be done.

Create a post-exit for the method

```
IF_USMD_CONV_SOM_GOV_ENTITY~WRITE_ENTITY_DATA(CL_USMD_CONV_SOM_GOV_API)
```

The relevant coding depends on the rules. For this example, you can find the code in chapter 8.1.5.

Add method to the Post-Exit.

A quick guide on how to enhance methods can be found on the SDN. The following document follows this approach:

8.1.3 Prerequisite: Create Z-Class

Create a Z-class, for example Z_CL_ENFORCE_BRF_PLUS_CALL, containing a static/public method, for example ENFORCE_BRF_PLUS_CALL.

The method needs to provide the following:

1. Input parameters

<table>
<thead>
<tr>
<th>Ty</th>
<th>Parameter</th>
<th>Type spec</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>IV_MODEL</td>
<td>TYPE USMD_MODEL</td>
<td>Data Model</td>
</tr>
<tr>
<td></td>
<td>IT_ENTITY_DATA</td>
<td>TYPE USMD_GOV_API_TS_ENT_DATA</td>
<td>Data referring to the source entity</td>
</tr>
<tr>
<td></td>
<td>IV_SOURCE_ENTITY</td>
<td>TYPE USMD_ENTITY</td>
<td>Entity Type</td>
</tr>
<tr>
<td></td>
<td>IT_SOURCE_ATTR</td>
<td>TYPE USMD_T_FIELDNAME</td>
<td>Attribute of the source entity</td>
</tr>
<tr>
<td></td>
<td>IV_TARGET_ENTITY</td>
<td>TYPE USMD_ENTITY</td>
<td>Target entity, for which BRF+ rule will be called</td>
</tr>
<tr>
<td></td>
<td>IV_TARGET_KEY</td>
<td>TYPE USMD_T_FIELDNAME</td>
<td>Field Names of keys</td>
</tr>
<tr>
<td></td>
<td>IV_USMD_STRUCT</td>
<td>TYPE USMD_STRUCT_OPTIONAL</td>
<td>Usmd structure type</td>
</tr>
<tr>
<td></td>
<td>IO_CORE_OBJECT</td>
<td>TYPE REF TO CL_USMD_CONV_SOM_GOV_API</td>
<td>MD Governance API for Single Object Maint. (Convenience)</td>
</tr>
</tbody>
</table>

2. Method code

```plaintext
*********************************************************************
* Method enforces the BRF+ rules for the target entity, if the value of *
* the source attribute is changed.                                  
*********************************************************************

data:
ls_entity_data      type usmd_gov_api_s_ent_data,
lt_entity_keys      type usmd_gov_api_ts_ent_tabl,
ls_entity_keys      type usmd_gov_api_s_ent_tabl,
lt_changes          type usmd_t_changed_entities,
ls_changes          type usmd_s_changed_entities,
lo_model            type ref to if_usmd_model_ext,
lo_so_gov_api       type ref to if_usmd_conv_som_gov_api,
```
lv_model type usmd_model,
lt_data type ref to data,
lo_instance type ref to if_usmd_model,
lo_app_context type ref to if_usmd_app_context,
lv_edition type usmd_edition,
lv_entity type usmd_entity,
lv_entity_fieldname type usmd_fieldname,
lt_entity_fieldname type usmd_ts_entity_fieldname,
lv_fieldname type usmd_fieldname,
l_entity_fieldname type usmd_ts_entity_fieldname,
ls_entity_fieldname type usmd_s_entity_fieldname,
l_entity_fieldname type usmd_ts_entity_fieldname,
lt_data type usmd_gov_api_ts_ent_data,
ls_data type usmd_gov_api_s_ent_data,
lv_enforce type boole_d,
lt_attr type usmd_ts_field,
l_attr type usmd_s_field,
lt_attr type usmd_attribute,
l_attr type usmd_s_field,
l_sel type usmd_ts_sel,
l_attr type usmd_s_sel,
lv_lines type int4,
lv_usmd_value type usmd_value,
lv_usmd_struc type usmd_struct,
l_message type usmd_t_message.

field-symbols:
<lt_entity_data> type index table,
<ls_entity_data> type any,
<ls_changed_entity> type usmd_s_changed_entity,
<ls_key_value> type usmd_s_attr_value,
<lt_data> type sorted table,
<ls_data> type any,
<lv_value> type any.

if iv_source_entity is initial or it_source_attr is initial or
   iv_target_entity is initial.
   return.
endif.

"------------------------------------------------------------------
  " Preparation
"------------------------------------------------------------------
" Determine the application context and the change request of the
" context.
lo_app_context = cl_usmd_app_context=>get_context( ).
"Get model-instance and the edition
if lo_app_context is bound.
   lo_app_context=>get_attributes( 
      importing
ev_edition = lv_edition
eo_model = lo_model ).
assert lo_model is bound.
   lv_model = lo_model->if_usmd_model_metadata_ext->d_usmd_model.
assert lv_model is not initial.
endif.

" Belong the BRF_plus-rule to the right model? If not return.
if iv_model <> lv_model.
    return.
endif.

" Get a suitable modelinstance to get the field name of the entity
" (the previous calculated model-instance cannot be needed)
cl_usmd_model=>get_instance(
    exporting
        i_usmd_model = lv_model
    importing
        eo_instance = lo_instance
        et_message = lt_message ).
assert lo_instance is not initial.

"------------------------------------------------------------------

" Checks
"------------------------------------------------------------------

"Get fieldname of the source entity
lv_entity_fieldname = cl_usmd_services=>entity2fieldname(
    io_model = lo_instance
    i_entity = iv_source_entity ).

lv_entity = lv_entity_fieldname.

" Do entity data refer to the source entity?
loop at it_entity_data into ls_entity_data where entity = lv_entity.
    clear ls_entity_keys.
    ls_entity_keys-entity = lv_entity.
    ls_entity_keys-tabl = ls_entity_data-entity_data.
    insert ls_entity_keys into table lt_entity_keys.
endloop.

" If the entity data do not refer to the source entity => don't
" enforce a rule
if lt_entity_keys is initial.
    return.
endif.

" Check whether an attribute from the given list IT_SOURCE_ATTR
" has been changed: If yes then enforce the BRF_plus rule for the
" target entity, otherwise return...

" Get changed attributes of the source entity
call method io_core_object->get_entity_field_changes
    exporting
        iv_struct = 'KATTR'
        it_entity_keys = lt_entity_keys
iv_contained_changes = abap_false
receiving
rt_changes = lt_changes.

" Check
loop at lt_changes into ls_changes where entity_type = lv_entity.
  if lv_enforce = abap_true.
    exit.
  else.
    loop at it_source_attr into lv_fieldname.
      if lv_enforce = abap_true.
        exit.
      else.
        loop at ls_changes-changed_entities assigning <ls_changed_entity>.
          read table <ls_changed_entity>-changed_attributes with key fieldname = lv_fieldname transporting no fields.
          if sy-subrc = 0.
            lv_enforce = abap_true.
            exit.
          endif.
        endloop.
      endif.
    endloop.
  endif.
endloop.

" No attribute from the given list has been changed=> return
if lv_enforce = abap_false.
  return.
endif.

"---------------------------------------------------------------
" Enforce rule
"---------------------------------------------------------------

" Get fieldname of the target entity
clear lv_entity_fieldname.
lv_entity_fieldname = cl_usmd_services=>entity2fieldname(
  io_model = lo_instance
  i_entity = iv_target_entity).
if lv_entity_fieldname is initial.
  return.
endif.

" Get reference for target entity data
if iv_usmd_struc is supplied.
  lv_usmd_struc = iv_usmd_struc.
else.
lv_usmd_struc = if_usmd_model_ext=>gc_struct_key_attr.
endif.
lo_model->create_data_reference(
  exporting
    i_fieldname = lv_entity_fieldname
    i_struct = lv_usmd_struc
    if_table = abap_true
    i_tabtype = if_usmd_model_ext=>gc_tabtype_sorted
  importing
    er_data = ltr_data ).
assert ltr_data is bound.
assign ltr_data->* to <lt_data>.

"Selection options
clear lt_sel.
loop at it_entity_data into ls_entity_data where entity = lv_entity.
  assign ls_entity_data-entity_data-* to <lt_entity_data>.
  loop at <lt_entity_data> assigning <ls_entity_data>.
    loop at it_target_key into lv_fieldname.
      assign component lv_fieldname of structure <ls_entity_data> to <lv_value>.
      if sy-subrc = 0.
        clear ls_sel.
        ls_sel-sign = 'I'.
        ls_sel-option = 'EQ'.
        ls_sel-fieldname = lv_fieldname.
        ls_sel-low = <lv_value>.
        insert ls_sel into table lt_sel.
      endif.
    endloop.
  endloop.
endloop.

" Add edition into it_sel (it is not necessary for material/BP)
if lv_edition is not initial.
  clear ls_sel.
  ls_sel-sign = 'I'.
  ls_sel-option = 'EQ'.
  ls_sel-fieldname = usmd0_cs_fld-edition.
  ls_sel-low = lv_edition.
  insert ls_sel into table lt_sel.
endif.

" Read target entity data
lo_model->read_char_value(
  exporting
    i_fieldname = lv_entity_fieldname
    it_sel = lt_sel
  importing
    et_data = <lt_data> ).
if <lt_data> is not initial.
ls_data-entity = lv_entity_fieldname.
ls_data-entity_data = ltr_data.
insert ls_data into table lt_data.

" Enforce call of the BRF+ rule for the target entity...a write
" is enough!
call method io_core_object->write_entity_data
    exporting
        it_entity_data = lt_data.
endif.

8.1.4 Create Derivation for MARCMRPPP

Open transaction USMD_RULE and create a derivation with the name *DERIVE_MARCMRPPP*. Right click on the node *DERIVATION* and navigate to *Create Function* as shown below.
How To... Check and Derivation Rules for MDG Material

Enter the name `DERIVE_MARCMRPPP`, a Short Text, and a Text.

Choose `Create` and accept your decision to get the following screen. Click on `Edit` and Add Existing Data Object.
Search for structure MARCMRPPP and choose OK.

Create the ruleset: Choose Assigned Rulesets and then choose Create Ruleset as shown on the following screen.
Enter a Name, a Short Text and a Text. Choose Create And Navigate To Object. You need a table containing the key-values of the source entity so create a variable, TABLE_OF_KEY_VALUES, of type Table by choosing the icon to the right of the field <Not assigned> and navigate to Create as shown below.

On the screen select the Type Table:
You get the following screen:

Enter a Name, for example Table of keys/values, a Short Text, and a Text. Enter in the field Binding Type the value Binding to DDIC Table (Data Dictionary) to get the following screen.
Maintain *USMD_T_ATTR_VALUE* in the field *DDIC Table Type* and choose *OK* to get the following screen:

Choose *Create And Navigate To Object*. Choose *OK* to get the following screen:
How To... Check and Derivation Rules for MDG Material

Activate, choose Back, and accept your decision to get the following screen:

Create the first rule. Choose Insert Rule and navigate to Create. Enter a Description, for example Create Table, choose the first Add (then-branch), and navigate to More.
On the obtained screen, navigate to the first component \textit{ATTR} of the structure \textit{USMD_S_ATTR_VALUE}.

Choose \textit{OK} to get the following screen.
Maintain the value `MATERIAL` in the empty box, choose `Options`, and navigate to `More`. Select the second component `VALUE` of the structure `USMD_S_ATTR_VALUE` and choose `OK` to get the following screen:

Choose `Change` and navigate to `MATERIAL` as shown on the following screen:
How To... Check and Derivation Rules for MDG Material

Choose **Options** and navigate to **More** as shown below:
On the next screen select the object *Table_for_key_values* and choose OK.

**Edit Rule**

If Assign Condition...  

Then

1. Change value of *USMD_S_A..ATTR* to *MATERIAL*
2. Change *USMD_S_A..VALUE* from value of *MARCMRPPP..MATERIAL*
3. Change *TABLE_FOR_KEY_VALUES* from value of *<assigned>*

Else  

Add
Choose Change and select the entry Insert as shown on the following screen:

You get the following screen:

Choose the icon next to <Not assigned> and select the structure USMD_S_ATTR_VALUE.
Choose Options and in the same way as before, create a new entry of the structure USMD_S_ATTR_VALUE with ATTR = WERKS and the value MARCMRPPP-WERKS. Then insert this entry into the table TABLE_FOR_KEY_VALUES to get the following result:
How To... Check and Derivation Rules for MDG Material

Choose OK and Activate and accept your decisions.
Create the second rule. Choose Options and choose Insert Rule and select Create.
On the next screen choose *Assigned Condition* and navigate to *Create Expression* as shown below.

Enter a *Description*, for example *Get_attribute value*. Choose Assign Condition and navigate to *Create Expression* and select type “Procedure Call” to get the following screen:
On the next screen enter a name, for example *Get attribute MMSTA*, a *Short Text* and a *Text*, as shown below.

Choose *Create And Navigate To Object* and choose *Yes* to get the following result:

Enter `CL_USMD_RULE_SERVICE` in the field *Class Name* and `GET_ATTR_VALUE` in the field *Method Name*. In the field *Result Data Object* assign the value *Attribute Value*: Choose the icon right
to <Not assigned> and choose the value ATTRIBUTE_VALUE_01 to get the following screen:

Choose “Add Parameter” and select all importing and exporting parameters one after the other. After these steps, you get the following result:
Choose the parameter *IV_ENTITY*, choose *<Not assigned>*, and enter the name of your source entity, in this case *MARC BASIC*.

Choose *OK*. Select the parameter *IT_KEY_VALUE* and then choose the icon right to *<Not assigned>* and navigate to *Select Context Parameter* as shown below:

Select the entry *TABLE FOR KEY_VALUE* to get the following screen:

Choose the parameter *IV_ATTR* and then choose *<Not assigned>*. On the next screen enter the value of your attribute, in this case *MMSTA*.
Choose OK to get the following screen:

Choose on the last parameter *EV_VALUE* and then choose the icon right to <Not Assigned> and select the entry *ATTRIBUTE_VALUE_01* as shown below:

After this step, you get the following screen. Activate and choose Yes.

Go back to the Ruleset. Make sure that the sequence of the rules is correct.
Choose **Edit** in the second rule to get the following screen:

Choose the icon right of the action **GET_ATTRIBUTE_MNSTA** and navigate to **Select Expression**, as shown below:
On the next screen, select your Object `GET_ATTRIBUTE_MMSTA` created before as shown below:
Choose OK to get the next screen:

Enter your value “40” in the empty field. Choose or and change it to and.
Choose the icon right of <Not assigned> and navigate to DISMM, as shown below:

Enter your value “ND” in the empty field. Then change the Active Template. Click on <1>and<2>.
How To... Check and Derivation Rules for MDG Material

Enhance the logic schema with value “and <3>”.

Add new condition stating that DGRMRPPP should be initial.
How To... Check and Derivation Rules for MDG Material

Choose the first Add (then-branch) and navigate to your attribute of the target entity, in this case DISMM as shown below:
Assign value ‘PD’. Then click on Options and to the same for DGRMRPPP and enter your value for it.

Enter your value, in this case ‘0001’, in the empty rectangle and choose OK.
Activate and then accept your decision. Choose Back and activate.

Go back and activate the function.
Now the BRF+ derivation with ruleset containing a rule to generate the key values table and a rule containing a procedure is available.

8.1.5 Add method to the Post-Exit

Run transaction SE24. Open class interface CL_USMD_CONV_SOM_GOV_API. Go to the Post-Exit in method IF_USMD_CONV_SOM_GOV_ENTITY~WRITE_ENTITY_DATA.

To enforce the BRF+ call for the target entity, add the following code:
How To... Check and Derivation Rules for MDG Material

**data:**

```
lv_field type usmd_fieldname,
lt_source_attr type usmd_t_fieldname,
lt_target_key type usmd_t_fieldname.
```

* Append the name of the corresponding source entity attribute on which the update of the target entity data depends

```
clear lt_source_attr.
lv_field = 'MMSTA'.
append lv_field to lt_source_attr.
```

* Append the keys of your target entity

```
clear lt_target_key.
lv_field = 'MATERIAL'.
append lv_field to lt_target_key.
lv_field = 'WERKS'.
append lv_field to lt_target_key.
```

```
Z_CL_ENFORCE_BRF_PLUS_CALL=>ENFORCE_BRF_PLUS_CALL(
    exporting
    iv_model = 'MM' "your model
    it_entity_data = it_entity_data
    iv_source_entity = 'MARCBASIC' "your source entity
    io_core_object = core_object
    it_source_attr = lt_source_attr
    iv_target_entity = 'MARCMRPPP' "your target entity
    it_target_key = lt_target_key
).
```

**Note:**

- For the creation of the procedure call Get attribute value you can use the method CHECK_ATTR_VALUE(CL_USMD_RULE_SERVICE) instead of the method GET_ATTR_VALUE(CL_USMD_RULE_SERVICE).
- Method IS_ENTITY_EMPTY(CL_USMD_RULE_SERVICE) can also be useful to check whether the data of an entity is empty.

**8.1.6 Test the BRF+ Rule**

Change an existing material. Go to the plant details. Open UIBB for Plant: General Data and Plant: Material Requirement Planning.
Check the existing values for Plant-Specific Material Status, MRP Group, and MRP Type.

Now, enter the value ‘40’ for Plant-Specific Material Status and press return. The result should be the following.
9. Cross-Entity Derivation with BAdI
USMD_RULE_SERVICE_CROSS_ET

The BAdI USMD_RULE_SERVICE_CROSS_ET is called at least once at every round trip if data was changed. Technical note: the BAdI is triggered in the Flush Method of the Governance API.

- Valid for Single Item Maintenance UI, Multi-Record Processing, application USMD_MASS_CHANGE and Data import
- Messages from derives are always converted into information messages.
- The BAdI gets only the attributes that are changed since the last BAdI call. Unchanged attributes are not transferred by the interface of this BAdI. You need to use the IO_MODEL parameter (method READ_CHAR_VALUE or RETRIEVE_CHAR_VALUE) to determine their values. For more information, see the documentation of interface IF_USMD_MODEL_EXT. Hint: Set parameter IF_NO_FLUSH to ABAP_TRUE to avoid additional calls of the Flush method.

Example Scenario:
You want to derive the material description from the characteristic valuation of the characteristics AAD268002 and AAD126067 (classification).

Example Solution:
SAP Notes 1806103 and 1815882 provide the example BAdI implementation MDG_BS_MAT_CLF2DESCRIPTION for the cross entity derive BAdI USMD_RULE_SERVICE_CROSS_ET.
SAP Note 1818020 is also required.

In SE18, enhancement spot USMD_RULE_SERVICE, copy the example implementation.
How To... Check and Derivation Rules for MDG Material

Change the characteristics in the attributes of the class interface to your values:

Example Characteristics:

Result on the material UI: The BAdI implementation concatenates two characteristic valuations to a new material description in two languages.
Limitation:
The material description is updated in the staging buffer and can be saved. But on the UI, the updated data is only displayed after the next roundtrip. That means if you change only the classification values the description is not immediately updated on the UI. You need to do another user action (such as choosing the Check button or changing other values) to see the updated data.

Technical background:
The reason for this behavior is that the classification UIBB (reuse component) is not embedded into the FPM phase model. The event CLF_CHANGED of Web Dynpro component /PLMU/WDC_CLF is raised in component controller method RETRIEVE_CLASSES_VALUES to inform consumers that need to synchronize data. This event is actual part of the Web Dynpro phase WDDOMODIFYVIEW that does not allow triggering an additional roundtrip in FPM. FPM needs to request an additional FPM roundtrip during processing of a Web Dynpro action handler method. Consumers of the Web Dynpro event for classification changed cannot trigger any FPM roundtrip in WDC phase modify view.
10. Technical environment BRF+ Landscape and Rule Maintenance

10.1 Use case
The application FMDM_MODEL_MM is a customizing object, client dependent, and transportable.

The Customer wants to edit the content of decision tables for Rules locally in the production system.

The definition of Validation and Derivation Rules (USMD_RULE) is customizing. Therefore, the maintenance depends on the client settings (SCC4 - Client Maintenance). However, some customers want to change transported BRF+ customizing objects in the production system independent from the change settings for the client customizing in transaction SCC4.

Solution:
Customer needs to decide what he wants to transport across the landscape OR transport once and change in the production system. A possible solution is provided by SAP Note 1462550. It describes how the change settings can be overwritten in an application exit (follow-up SAP Note 1501679 might be also relevant).

10.2 Example for Application Exit GET_CHANGEABILITY
The changeability exit allows overruling the client settings for the changeability of client-dependent objects. Also, the automatic recording of changes on transport requests can be adapted. However, when transportable customizing objects become editable in this way, changes may get lost with the next transport from a customizing system or a client copy.

See document BRF+ Application Exits: [http://scn.sap.com/docs/DOC-4564](http://scn.sap.com/docs/DOC-4564). In this document, you can also find methods for additional authorization checks.

IF_FDT_APPLICATION_SETTINGS
General Implementation
The interface IF_FDT_APPLICATION_SETTINGS defines a set of static methods – the actual Application Exits. For each method/exit a corresponding Boolean attribute GV_<method name> is defined by the interface. The attribute is used to indicate whether the interface-method is implemented and the exit shall be used. An exit-method is called by the BRF+ framework only, if the corresponding Boolean attribute has been set to true. Setting the attributes to true is best done inside a static class constructor, as in the following example:

```abap
METHOD class_constructor.
  IF_FDT_APPLICATION_SETTINGS~gv_authority_check = abap_true.
  IF_FDT_APPLICATION_SETTINGS~gv_get_element_values = abap_true.
ENDMETHOD.
```

Depending on the triggering event and the called method’s interface, the application exit may influence the behavior of BRF+’ object handling. Since the currently executed program exits the BRF+ at this point, it is in the responsibility of the programmer to avoid any kind of runtime errors.
METHOD if_fdt_application_settings~get_changeability.

DATA: l_expression_type_id TYPE if_fdt_types=>id.

*--Entire BRF+ application is allowed to be modified in Dev system.
IF sy-sysid+0(1) EQ 'D'.
  " Dev system
  cv_changeable = abap_true.
  RETURN.
ENDIF.

*--Import the expression type
cl_fdt_factory=>get_id_information( EXPORTING iv_id = iv_id
  IMPORTING ev_expression_type_id = l_expression_type_id ).

*--If expression type is decision table
if l_expression_type_id EQ if_fdt_constants=>gc_exty_decision_table.
  cv_changeable = abap_true.
ELSE.
  cv_changeable = abap_false.
ENDIF.
ENDMETHOD.