# Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1 - About This Guide</td>
<td>1</td>
</tr>
<tr>
<td>1.1. Related Publications</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2 - Purpose of the MetaStore Manager</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 3 - Key Notions</td>
<td>4</td>
</tr>
<tr>
<td>Chapter 4 - Prerequisites</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 5 - Getting Started</td>
<td>6</td>
</tr>
<tr>
<td>Chapter 6 - MetaStore Manager User Interface</td>
<td>8</td>
</tr>
<tr>
<td>6.1. Menu Bar</td>
<td>8</td>
</tr>
<tr>
<td>6.2. Toolbar</td>
<td>10</td>
</tr>
<tr>
<td>6.3. Tree View Window</td>
<td>11</td>
</tr>
<tr>
<td>6.4. Workspace</td>
<td>12</td>
</tr>
<tr>
<td>6.5. Record Fields Window</td>
<td>13</td>
</tr>
<tr>
<td>6.6. Output Window</td>
<td>15</td>
</tr>
<tr>
<td>6.7. Record Layout Window</td>
<td>16</td>
</tr>
<tr>
<td>6.8. Statusbar</td>
<td>17</td>
</tr>
<tr>
<td>6.9. Docking a Window</td>
<td>17</td>
</tr>
<tr>
<td>Chapter 7 - Creating Dictionary Files - Overview</td>
<td>19</td>
</tr>
<tr>
<td>Chapter 8 - Building Dictionary Files Manually - Overview</td>
<td>20</td>
</tr>
<tr>
<td>Chapter 9 - Adabas File Group</td>
<td>21</td>
</tr>
<tr>
<td>9.1. Creating the Dictionary File</td>
<td>21</td>
</tr>
<tr>
<td>Technical Tab (Adabas File Groups)</td>
<td>22</td>
</tr>
<tr>
<td>Business Tab (Adabas File Groups)</td>
<td>23</td>
</tr>
<tr>
<td>9.2. Defining Records</td>
<td>24</td>
</tr>
<tr>
<td>Technical Tab (Adabas Files)</td>
<td>24</td>
</tr>
<tr>
<td>Business Tab (Adabas Files)</td>
<td>27</td>
</tr>
</tbody>
</table>
9.3. Defining Fields................................................................................................................................. 28
   Technical Tab (Adabas Fields) .............................................................................................................. 28
   Business Tab (Adabas Fields) ............................................................................................................... 38
9.4. Defining Relationships...................................................................................................................... 38
   Technical Tab (Adabas Relationships) ................................................................................................. 39
   Business Tab (Adabas Relationships) .................................................................................................. 42
9.5. Defining Indices............................................................................................................................... 42
   Technical Tab (Adabas Indices) ............................................................................................................ 43
   Business Tab (Adabas Indices) ............................................................................................................ 45
9.6. Business Tab (Adabas Data Objects)............................................................................................... 45
   Business Rule .................................................................................................................................... 45
   Note..................................................................................................................................................... 45

Chapter 10 - Datacom File Group........................................................................................................... 46
10.1. Creating the Dictionary File............................................................................................................ 46
   Technical Tab (Datacom File Groups) .................................................................................................. 47
   Business Tab (Datacom File Groups) .................................................................................................. 48
10.2. Defining Records ............................................................................................................................ 49
   Technical Tab (Datacom Records) ....................................................................................................... 49
   Business Tab (Datacom Records) ....................................................................................................... 52
10.3. Defining Fields............................................................................................................................... 52
   Technical Tab (Datacom Fields) .......................................................................................................... 53
   Business Tab (Datacom Fields) .......................................................................................................... 63
10.4. Defining Relationships.................................................................................................................. 63
   Technical Tab (Datacom Relationships) ............................................................................................. 64
   Business Tab (Datacom Relationships) ............................................................................................. 67
10.5. Defining Indices............................................................................................................................. 67
   Technical Tab (Datacom Indices) ........................................................................................................ 68
   Business Tab (Datacom Indices) ........................................................................................................ 69
10.6. Business Tab (Datacom Data Objects)......................................................................................... 69
   Business Rule .................................................................................................................................... 70
   Note..................................................................................................................................................... 70

Chapter 11 - IMS PCB............................................................................................................................ 71
11.1. Creating the Dictionary File.......................................................................................................... 71
   Technical Tab (IMS PCB) ..................................................................................................................... 72
   Business Tab (IMS PCB) .................................................................................................................... 73
11.2. Defining Records.......................................................................................................................... 74
   Technical Tab (IMS Segments) ............................................................................................................ 74
   Business Tab (IMS Segments) ............................................................................................................ 77
11.3. Defining Fields.............................................................................................................................. 77
   Technical Tab (IMS Fields) ................................................................................................................ 78
   Business Tab (IMS Fields) ................................................................................................................ 87
### Chapter 12 - SQL Table Group

- **12.1. Creating the Dictionary File** ................................................................. 92
  - Technical Tab (SQL Table Groups) .............................................................. 93
  - Business Tab (SQL Table Groups) .............................................................. 96
- **12.2. Defining Records** ................................................................................ 97
  - Technical Tab (SQL Tables) ...................................................................... 97
  - Business Tab (SQL Tables) ...................................................................... 98
- **12.3. Defining Fields** ................................................................................... 99
  - Technical Tab (SQL Columns) ................................................................. 100
  - Business Tab (SQL Columns) ................................................................. 108
- **12.4. Business Tab (SQL Table Group Data Objects)** .................................. 108
  - Business Rule ......................................................................................... 108
  - Note ........................................................................................................ 108

### Chapter 13 - Standard File

  - Technical Tab (Standard Files) ............................................................... 110
  - Business Tab (Standard Files) ............................................................... 117
- **13.2. Defining Records** ................................................................................ 117
  - Technical Tab (Standard Records) .......................................................... 118
  - Business Tab (Standard Records) .......................................................... 121
- **13.3. Defining Fields** ................................................................................... 122
  - Technical Tab (Standard Fields) ............................................................. 123
  - Business Tab (Standard Fields) ............................................................. 133
- **13.4. Business Tab (Standard File Data Objects)** ....................................... 133
  - Business Rule ......................................................................................... 133
  - Note ........................................................................................................ 133

### Chapter 14 - IDMS Subschema

- **14.1. Creating the Dictionary File** ................................................................. 135
  - Technical Tab (Subschemas) ................................................................. 135
  - Business Tab (Subschemas) ................................................................. 137
- **14.2. Defining Records** ................................................................................ 137
  - Technical Tab (Subschema Records) ...................................................... 138
  - Business Tab (Subschema Records) ...................................................... 140
- **14.3. Defining Fields** ................................................................................... 140
  - Technical Tab (Subschema Elements) .................................................... 141
  - Business Tab (Subschema Elements) .................................................... 151
# Table of Contents

**IKAN Solutions METAVERSE METASTORE MANAGER - RELEASE 8.1.3**

1. **Processing the MXL File** ................................................. 209
2. **Chapter 17 - File Type: XML Schema** .......................... 211
   17.6. **File Type: XML Schema** .............................................. 211

3. **Chapter 18 - Target Dictionary Files** ............................... 212
   18.1. **File Type: Load Sequential** ........................................ 212
      Procedure ............................................................................... 212
      Description of the Available Fields ...................................... 219
   18.2. **File Type: Load Delimited** ......................................... 223
      Procedure ............................................................................... 224
      Description of the Available Fields ...................................... 227
      Load Delimited Options ......................................................... 230
   18.3. **Load RDBMS** ............................................................. 235
      Procedures ............................................................................. 235

4. **Chapter 19 - Importing MDL Files** .................................. 241

5. **Chapter 20 - Exporting Dictionary Files** ........................ 242
   20.1. **General Export Procedure** ........................................ 242

6. **Chapter 21 - Selecting another User Profile** ................... 244

7. **Chapter 22 - Version Management with Source Control** .... 245
   22.1. Establishing the Connection Between MetaStore and the Source Control System .............. 245
   22.2. Terminating the Connection Between MetaStore and the Source Control System .............. 246
   22.3. Adding Dictionary Files to Source Control ....................................................... 246
   22.4. Showing the Source Control Status of Opened Source Files ........................................ 247
   22.5. Performing Changes to Dictionary Files Under Source Control ........................................ 248
   22.6. Undoing the Check-Out of a Dictionary File ....................................................... 249

8. **Chapter 23 - Internal Source Control** .............................. 250
   23.1. Metadata Versioning Rules .............................................. 250
   23.2. Incrementing the Version Number of a Dictionary File .............................................. 251
   23.3. Creating a New Instance of a Dictionary File ....................................................... 252
   23.4. Opening a Specific Version of a Dictionary File ....................................................... 252

9. **Chapter 24 - Calling the MetaStore Manager in Batch** .... 254
   24.1. Using MSBSTORE to Import MDL Files into the MetaStore Dictionary ......................... 254
      Command format .................................................................. 254
      Example ............................................................................... 255
   24.2. Using MSBSTORE to Export MDL Files from the MetaStore Dictionary ......................... 255
      Command format .................................................................. 255
      Example ............................................................................... 255
   24.3. Using MSBSTORE to Collect Files ........................................ 256
Chapter 25 - Definition Language Commands ......................................................... 258

25.1. ADD FILE .................................................................................................................. 258
Format ................................................................................................................................. 258
ADD FILE .......................................................................................................................... 258
VERSION ............................................................................................................................. 259
TYPE .................................................................................................................................... 259
Record-size ......................................................................................................................... 261
BLOCK ................................................................................................................................. 262
SPANMED .......................................................................................................................... 262
MODE ................................................................................................................................... 262
LABEL ................................................................................................................................. 263
KEY ...................................................................................................................................... 263
CODE-CONTROL ............................................................................................................. 263
FETCH-FIRST-ONLY ......................................................................................................... 263
UNCOMMITTED-READ ..................................................................................................... 263
EXTERNAL-SOURCE .......................................................................................................... 264
COLUMN-SEPARATOR ....................................................................................................... 265
ROW-TERMINATOR .......................................................................................................... 265
CCSID ................................................................................................................................. 265
XPATH ................................................................................................................................. 266
XML-DECLARATION ......................................................................................................... 266
RULE ................................................................................................................................. 266
NOTE ................................................................................................................................. 266

25.2. ADD RECORD ............................................................................................................ 266
Format ................................................................................................................................. 267
ADD RECORD .................................................................................................................... 267
file-name ........................................................................................................................... 267
SIZE .................................................................................................................................. 267
KEY .................................................................................................................................. 267
COLUMN-SEPARATOR ...................................................................................................... 268
ROW-TERMINATOR .......................................................................................................... 269
CCSID ................................................................................................................................. 269
XPATH ................................................................................................................................. 269
RULE ................................................................................................................................. 270
NOTE ................................................................................................................................. 270

25.3. ADD FIELD ................................................................................................................ 270
Format ................................................................................................................................. 270
ADD FIELD ....................................................................................................................... 270
POSITION ......................................................................................................................... 271
SIZE ................................................................................................................................. 271
# Detailed Table of Contents

- **OCCURS** .................................................................................................................. 272
- **DEPENDING ON** ........................................................................................................ 273
- **TYPE** .......................................................................................................................... 273
- **DATE** .......................................................................................................................... 277
- **TIME** .......................................................................................................................... 278
- **TIMESTAMP** ............................................................................................................... 278
- **EDIT** ........................................................................................................................... 278
- **LIMITS** ....................................................................................................................... 281
- **NULL-INDICATOR (former DBNAME)** ......................................................................... 281
- **INITIAL** ...................................................................................................................... 283
- **XML-NAME** ............................................................................................................... 283
- **XML-TYPE** ................................................................................................................ 283
- **FORMAT-MASK** ......................................................................................................... 283
- **CCSID** ......................................................................................................................... 283
- **XPATH** ......................................................................................................................... 283
- **RULE** .......................................................................................................................... 283
- **NOTE** .......................................................................................................................... 284

## 25.4. COPY FILE

- **Format** ..................................................................................................................... 284

## 25.5. COPY RECORD

- **Format** ..................................................................................................................... 284

## 25.6. COPY FIELD

- **Format** ..................................................................................................................... 285

## 25.7. DELETE FILE

- **Format** ..................................................................................................................... 285

## 25.8. LIST FILE

- **Format** ..................................................................................................................... 285

## 25.9. LIST RECORD

- **Format** ..................................................................................................................... 286

## 25.10. LIST FIELD

- **Format** ..................................................................................................................... 286
The MetaStore Manager User Guide is intended for MetaSuite metadata administrators.

1.1. Related Publications
The following table gives an overview of the complete MetaSuite documentation set.

<table>
<thead>
<tr>
<th>Release Information</th>
<th>Release Notes 8.1.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation Guides</td>
<td>• BS2000/OSD Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• DOS/VSE Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• Fujitsu Windows Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• MicroFocus Windows Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• MicroFocus UNIX Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• OS/390 and Z/OS Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• OS/400 Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• VisualAge Windows Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• VisualAge UNIX Runtime Component</td>
</tr>
<tr>
<td></td>
<td>• VMS Runtime Component</td>
</tr>
<tr>
<td>User Guides</td>
<td>• INI Manager User Guide</td>
</tr>
<tr>
<td></td>
<td>• Installation and Setup Guide</td>
</tr>
<tr>
<td></td>
<td>• Introduction Guide</td>
</tr>
<tr>
<td></td>
<td>• MetaStore Manager User Guide</td>
</tr>
<tr>
<td></td>
<td>• MetaMap Manager User Guide</td>
</tr>
<tr>
<td></td>
<td>• Generator Manager User Guide</td>
</tr>
<tr>
<td>Technical Guides</td>
<td>• ADABAS File Access Guide</td>
</tr>
<tr>
<td></td>
<td>• IDMS File Access Guide</td>
</tr>
<tr>
<td></td>
<td>• IMS DLI File Access Guide</td>
</tr>
<tr>
<td></td>
<td>• RDBMS File Access Guide</td>
</tr>
<tr>
<td></td>
<td>• XML File Access Guide</td>
</tr>
<tr>
<td></td>
<td>• Runtime Modules</td>
</tr>
<tr>
<td></td>
<td>• User-defined Functions User Guide</td>
</tr>
</tbody>
</table>

If you are unfamiliar with MetaSuite, the following technical description provides you with a brief overview.

**The MetaSuite System**

MetaSuite is designed for data retrieval, extraction, conversion and reporting. It includes a workstation-based graphical user interface and a mainframe runtime component.
MetaSuite Database Interfaces
MetaSuite can access data from a number of database management systems, using the same commands, program structure and retrieval techniques used for non-database files. Each database interface is available as an optional enhancement to the base product.

MetaMap Manager
MetaMap Manager is the MetaSuite tool used to define models. Such models are intuitively built by describing overall program specifications, input file definitions (data and process) and target file definitions (data and process).

MetaStore Manager
MetaStore Manager is a tool that provides metadata maintenance and documentation services.

Generator Manager
The Generator Manager is the system administration tool. All kinds of basic functionalities and customization possibilities are supported by this tool.
MetaSuite is a data integration application that enables you to rapidly move large volumes of data from any Source to any Target Business Intelligence Environment. For this purpose, MetaSuite needs to know the metadata of all possible data sources and targets.

MetaSuite MetaStore Manager is responsible for providing these metadata and allows to:

- Create Dictionary Files automatically (page 180) or manually (page 20)
- Edit (or enrich) available metadata.
- Save Dictionary Files into the MetaStore Repository.
- Export Dictionary Files in MDL, XML, XMI or PDL format. See Exporting Dictionary Files on page 242.
### Key Notions

The following table explains the key notions used by the MetaStore Manager.

<table>
<thead>
<tr>
<th>Notion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MetaStore</td>
<td>MetaStore is the repository containing all Dictionary Files.</td>
</tr>
<tr>
<td>Dictionary File</td>
<td>A Dictionary File describes the metadata for the logical unit of data either from which data will be extracted (data source) or to which data will be written (data target). The metadata describe both the physical and business characteristics of the Dictionary File. A Dictionary File contains subordinate objects, such as the underlying Records, Fields, Relationships and Indices. Each Dictionary File contains one or more Records.</td>
</tr>
<tr>
<td>Record</td>
<td>A Record in a Dictionary File describes the metadata of a specific data structure within the Dictionary File. This structure is described by the Record's underlying Fields. Each Record contains one or more Fields.</td>
</tr>
<tr>
<td>Field</td>
<td>A Field is the smallest data unit that can be addressed within a Record. It has an internal datatype and a size.</td>
</tr>
<tr>
<td>Group Field</td>
<td>A Group Field is a Field that can be expanded in one or more Subfields.</td>
</tr>
<tr>
<td>Subfield</td>
<td>A Subfield is a Field that is defined as a component of another Field (Group Field).</td>
</tr>
<tr>
<td>Relationship</td>
<td>A Relationship defines a logical relation between two or more Records.</td>
</tr>
<tr>
<td>Index</td>
<td>An Index defines an access path to a Record.</td>
</tr>
</tbody>
</table>
The following prerequisites must be met before you can use the MetaStore Manager.

- Install the MetaSuite program
- Create a Repository
- Create ODBC access to the Repository
- Complete the setup after installation

If you will use a Version Management tool, the following additional steps need to be performed:

- Install this Version Management tool
- Create a repository for MetaSuite in this version management tool
- Allow user access on this repository
- Create a project within this repository

**Note:** All procedures you need to follow to obtain this situation are described in the *Installation and Setup Guide*. 
1. **Start the MetaStore Manager.**
   
   MetaStore Manager requires initial settings defined in an INI file. The default name of this INI file is `MetaSuite.ini` and its default location is the user's `AppData\Roaming\MetaSuite` folder.

   **Note:** If the `MetaSuite.ini` is not available at the expected location, the installation procedure will start automatically and the default `MetaSuite.ini` file will be created.

2. **The MetaStore Manager Logon screen appears:**

   ![](image.png)

   Fill out the fields as required:
Note: The default values that may be present in the field match the ones defined in the MetaSuite.ini file using the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Installation and Setup Guide.

3. Click OK.
The MetaStore Manager opening screen appears:
CHAPTER 6

MetaStore Manager User Interface

The MetaStore Manager User Interface is composed of the following elements:

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Menu Bar</strong> (page 8)</td>
<td>The Menu Bar gives access to the different MetaStore Manager functionalities.</td>
</tr>
<tr>
<td><strong>Toolbar</strong> (page 10)</td>
<td>The Toolbar gives access to frequently used functionalities, that can also be</td>
</tr>
<tr>
<td></td>
<td>accessed from the Menu Bar.</td>
</tr>
<tr>
<td><strong>Tree View Window</strong> (page 11)</td>
<td>The Tree View Window displays all opened Dictionary Files with their Records,</td>
</tr>
<tr>
<td></td>
<td>Fields, Relationships and Indices. If you just started working, the MetaStore</td>
</tr>
<tr>
<td></td>
<td>Root icon is the only icon available.</td>
</tr>
<tr>
<td><strong>Workspace</strong> (page 12)</td>
<td>In the Workspace, the Properties Panels for the MetaStore Objects are displayed. The Properties Panels can be used for verifying or updating the Properties.</td>
</tr>
<tr>
<td><strong>Record Fields Window</strong> (page</td>
<td>In this window, the Fields belonging to the Record that was last opened in the</td>
</tr>
<tr>
<td>13)</td>
<td>Workspace, are displayed.</td>
</tr>
<tr>
<td><strong>Output Window</strong> (page 15)</td>
<td>This window contains messages about all actions that took place while working</td>
</tr>
<tr>
<td></td>
<td>with the current session of MetaStore Manager.</td>
</tr>
<tr>
<td><strong>Record Layout Window</strong> (page</td>
<td>In this window, the layout of the last opened Dictionary Record is displayed.</td>
</tr>
<tr>
<td>16)</td>
<td></td>
</tr>
<tr>
<td><strong>Statusbar</strong> (page 17)</td>
<td>The Statusbar displays timestamps of the creation and last update of the selected MetaStore object (MetaStore Repository, Dictionary File, Record, etc.).</td>
</tr>
</tbody>
</table>

## 6.1. Menu Bar

Once you have started the MetaStore Manager, the Menu Bar is displayed at the top of the opening screen. It contains the following menus:
<table>
<thead>
<tr>
<th>Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>File</strong></td>
<td>The File menu contains the following options:</td>
</tr>
<tr>
<td>• <strong>Add</strong> (with submenu)</td>
<td>See <a href="#">Building Dictionary Files Manually - Overview</a> on page 20.</td>
</tr>
<tr>
<td>• <strong>Open/Delete File</strong>...</td>
<td>Use this option to open or to delete an existing Dictionary File from a list of available Dictionary Files.</td>
</tr>
<tr>
<td>• <strong>Collect File</strong>...</td>
<td>See <a href="#">Collecting Dictionary Files</a> on page 180.</td>
</tr>
<tr>
<td>• <strong>Import File</strong>...</td>
<td>See <a href="#">Importing MDL Files</a> on page 241.</td>
</tr>
<tr>
<td>• <strong>Save Properties</strong></td>
<td>Use this option to save the Properties of newly defined MetaStore objects or to save the new settings, if you modified the Properties of an existing MetaStore object.</td>
</tr>
<tr>
<td>• <strong>Save to MetaStore</strong></td>
<td>Use this option to save the changes you performed on the Dictionary File selected in the Tree View Window. This option is only active if you select a File Definition in the Tree View Window.</td>
</tr>
<tr>
<td>• <strong>Remove all Files from Workspace</strong></td>
<td>Use this option to remove all opened Dictionary Files from the Tree View Window. This action does not delete these Dictionary Files from the Repository. However, if the Tree View Window contained unsaved items, they will be lost.</td>
</tr>
<tr>
<td>• <strong>Export</strong> (with submenu)</td>
<td>See <a href="#">Exporting Dictionary Files</a> on page 242.</td>
</tr>
<tr>
<td>• <strong>Print Tree</strong>...</td>
<td>Use this option to print the MetaStore Tree as it is currently displayed in the Tree View Window.</td>
</tr>
<tr>
<td>• <strong>Reconnect</strong>...</td>
<td>Use this option to connect to another Repository. The connection to the current Repository is terminated and the MetaSuite Logon screen is displayed again.</td>
</tr>
<tr>
<td>• <strong>Exit</strong></td>
<td>Use this option to leave the program. If there are any unsaved changes to the MetaStore, you will be asked if you want to save them now.</td>
</tr>
<tr>
<td><strong>Edit</strong></td>
<td>The Edit menu contains the following standard Windows options:</td>
</tr>
<tr>
<td>• <strong>Undo</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Cut</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Copy</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Paste</strong></td>
<td>You can use these options to cut, copy and paste text entries in opened properties panels. You can also undo one of these actions.</td>
</tr>
<tr>
<td><strong>View</strong></td>
<td>The View menu contains the following options:</td>
</tr>
<tr>
<td>• <strong>Toolbar</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Statusbar</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Tree View</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Record Fields</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Record Layout</strong></td>
<td></td>
</tr>
<tr>
<td>• <strong>Output</strong></td>
<td>By default, all options are checked, meaning that all listed items are displayed on the screen.</td>
</tr>
</tbody>
</table>
6.2. Toolbar

If the Toolbar option in the View menu is checked, the Toolbar is displayed underneath the Menu Bar. The Toolbar contains the following buttons:

<table>
<thead>
<tr>
<th>Button</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add" /></td>
<td>Add (giving access to submenu)</td>
<td>You need to add or build a Dictionary File manually, if the metadata cannot be collected or imported. See Building Dictionary Files Manually - Overview on page 20.</td>
</tr>
<tr>
<td><img src="image" alt="Open/Delete" /></td>
<td>Open/Delete File...</td>
<td>You can use this option to open or delete Dictionary Files that are available in the MetaStore. You need to open an existing Dictionary File in order to verify or modify its Properties.</td>
</tr>
<tr>
<td><img src="image" alt="Collect" /></td>
<td>Collect File...</td>
<td>You need to collect a Dictionary File, if it does not yet exist in the MetaStore. The metadata can however be delivered in a standard format to the MetaStore. See Collecting Dictionary Files on page 180.</td>
</tr>
<tr>
<td><img src="image" alt="Import" /></td>
<td>Import File...</td>
<td>You need to import an MDL File, if the matching Dictionary File does not yet exist in the current MetaStore. See Importing MDL Files on page 241.</td>
</tr>
<tr>
<td><img src="image" alt="Remove all Files from Workspace" /></td>
<td>Remove all Files from Workspace</td>
<td>Use this option to remove all opened Dictionary Files from Workspace. This does not delete them from the MetaStore.</td>
</tr>
</tbody>
</table>
6.3. Tree View Window

If the Tree View option in the View menu is checked, the Tree View Window is displayed in the upper left corner.

**Note:** As this is a dockable window, you can modify its position ([Docking a Window](page 17)). You can also hide the window, by clicking the Auto Hide ( ![ ] ) icon in its upper right corner. Reclicking the Auto Hide ( ![ ] ) icon will restore the window to its original position.

Expand the different levels by clicking the plus sign next to the item. The following icons are used:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>Dictionary File. Each Dictionary File is listed by its name and by its version number. Each Dictionary File contains the metadata pertaining to a specific data source or data target.</td>
</tr>
<tr>
<td>![ ]</td>
<td>Record</td>
</tr>
<tr>
<td>![ ]</td>
<td>Relationship</td>
</tr>
</tbody>
</table>
### 6.4. Workspace

In a standard configuration, the Workspace is the grey zone next to the Tree View Window.

This area is used to display the properties panels.
6.5. Record Fields Window

In a standard configuration, the Record Fields Window is docked underneath the Workspace. It contains information about the Fields belonging to the Record, that was last opened in the Workspace.

You open a Record in the Workspace by double-clicking it or by selecting Properties from its context menu.

**Note:** As this is a dockable window, you can modify its position (Docking a Window (page 17)). You can also hide the window, by clicking the Auto Hide ( ) icon in its upper right corner. Reclicking the Auto Hide ( ) icon will restore the window to its original position.

The following figure shows an example of a Record Fields window.

The main Properties for each Field in the Record Fields window are displayed:

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>This field contains the name of the Field, as defined when it was added.</td>
</tr>
<tr>
<td>Type</td>
<td>This field contains the Field Type.</td>
</tr>
<tr>
<td>Abs. Position</td>
<td>If required by the Dictionary File type, this field contains the starting position for this field. If the starting position is not required by the Dictionary File type, this field remains empty.</td>
</tr>
<tr>
<td>Size</td>
<td>This field contains the size for this field.</td>
</tr>
<tr>
<td>Decimals</td>
<td>If the field type allows the definition of decimals, this field contains the number of decimals. If the field type does not allow the definition of decimals, this field contains the default value 0.</td>
</tr>
<tr>
<td>Date Format</td>
<td>If the field describes a date, the selected date format is displayed. If the field does not describe a date, the default message NONE is displayed.</td>
</tr>
<tr>
<td>Occurs</td>
<td>This field indicates the number of times this field occurs in the Record.</td>
</tr>
<tr>
<td>Null</td>
<td>This field indicates the nullable status of the field, and whether Inbound or Outbound nulls are used.</td>
</tr>
</tbody>
</table>

**Context Menu**

Right-click in the Record Fields Window.
The following context menu is displayed:

Note: One or more options may not be accessible, depending on the current settings or selection.

These options have the following meaning:

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert Field</td>
<td>Select this option to add a field to the current record above the currently selected field. If no field is currently selected, the new field will be added at the end of the record.</td>
</tr>
<tr>
<td>Add Subfield</td>
<td>Select this option to add a Subfield to the selected Field.</td>
</tr>
<tr>
<td>Append Field</td>
<td>Select this option to add a field at the end of the Record.</td>
</tr>
<tr>
<td>Delete Field</td>
<td>Select this option to delete the currently selected field from the record.</td>
</tr>
<tr>
<td>Field Properties</td>
<td>Select this option to display the Properties of the selected Field. If no field is selected, this option is not accessible.</td>
</tr>
</tbody>
</table>
| Auto Position       | Select this option if you want the position of the field to be recalculated automatically. Flagging this option makes it possible to automatically calculate the position. In most cases this results in the end position of the last field +1. There are some exceptions to this standard rule:  
  • The first field has no predecessor. Obviously, in this case the position will be 1.  
  • Redefines: in this case the position of the “redefined” field will be taken.  
  • Subfields: this is in fact a variation of redefines. The first subfield will have the same position as the group field it belongs to. The second subfield will follow the standard rule. |
| Auto Size           | Select this option if you want to automatically calculate the total field size of a group field. |
| Redefined Field     | Flags a field as having redefines (fields that are starting at the same position, usually redefining the field as another data type) |
| Redefines Field     | Locks the field's starting position to the position of those marked as 'redefined' (the step before). There can be several occurrences of 'redefine'. |
| FILLER field        | A FILLER field always uses ‘auto position’ meaning that it follows the previous field(s) when their position/size changes (a size filler). |
6.6. Output Window

If the *Output Window* option in the *View* menu is checked, the Output Window is by default displayed in the bottom left corner of the screen.

**Note:** As this is a dockable window, you can modify its position ([Docking a Window](page 17)). You can also hide the window, by clicking the *Auto Hide* ( ) icon in its upper right corner. Reclicking the *Auto Hide* ( ) icon will restore the window to its original position.

The *Output Window* contains messages and warnings pertaining to actions that have been performed during the current MetaSuite MetaStore Manager session.

### Context Menu

Right-click in the Output Window. The following context menu is displayed:

These options have the following meaning:

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear All</td>
<td>Select this option to clear all messages from the Output Window.</td>
</tr>
<tr>
<td>Copy All</td>
<td>Select this option to select all messages in the Output Window.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Layout</td>
<td>Only available for complex fields and records (field containers): it orders the contained fields sequentially, by setting their “Auto position” and “auto size” to ON.</td>
</tr>
<tr>
<td>Manual Layout</td>
<td>Select this option to turn off the auto flags of contained fields and to keep the current values.</td>
</tr>
</tbody>
</table>
6.7. Record Layout Window

If the Record Layout option in the View menu is checked, the Record Layout Window is by default displayed in the bottom right corner of the screen.

**Note:** As this is a dockable window, you can modify its position (Docking a Window (page 17)). You can also hide the window, by clicking the Auto Hide ( ) icon in its upper right corner. Reclicking the Auto Hide ( ) icon will restore the window to its original position.

The Record Layout window shows the following information about the last opened Record:

The Record Layout window shows a default value for each Field within the selected Record, dependent on the Field type. The default value matches the size of the Field, as defined in the Record Fields Window. Default values are used to represent the values in different Field types:

<table>
<thead>
<tr>
<th>Default Value</th>
<th>Field Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alphabetic</td>
</tr>
<tr>
<td>B</td>
<td>Binary</td>
</tr>
<tr>
<td></td>
<td>Binary native</td>
</tr>
<tr>
<td>1</td>
<td>Bit</td>
</tr>
<tr>
<td>b</td>
<td>Byte</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
<tr>
<td>D</td>
<td>Decimal</td>
</tr>
<tr>
<td>F</td>
<td>Float</td>
</tr>
<tr>
<td>G</td>
<td>Graphic</td>
</tr>
<tr>
<td></td>
<td>Vargraphic</td>
</tr>
<tr>
<td>H</td>
<td>Hexadecimal</td>
</tr>
<tr>
<td>?</td>
<td>Longvarchar</td>
</tr>
<tr>
<td></td>
<td>Longvargraphic</td>
</tr>
<tr>
<td></td>
<td>(not supported yet)</td>
</tr>
</tbody>
</table>
### 6.8. Statusbar

If the Statusbar option in the View menu is checked, the Statusbar is displayed in the bottom right corner of the screen.

If an item (the MetaSuite Repository, a Dictionary File, a Record or a Field) is selected in the Tree View Window, the Statusbar contains the following information for the selected item:

- Location of the INI file
- Creation user name and timestamp
- Last Update user name and timestamp
- NUM (Num Lock selected)

### 6.9. Docking a Window

Dockable windows are windows that align themselves with the edge of another interface element, another window or properties panel.

1. Click the window title bar and keep the mouse button pressed.
2. Drag the selected window to the required position.

The window you are repositioning is displayed in grey and positioning anchors are displayed on the screen.
3. Place the cursor on the anchor of your choice, and release the mouse button.

**Note:** You can also position the window outside the main MetaSuite window, on your Desktop. To return the window to its previous position within the MetaSuite window, double-click its header.
# CHAPTER 7

## Creating Dictionary Files - Overview

There are three ways to create Dictionary Files. They all match a command in the File Menu on the MetaStore Menu Bar, and an icon on the MetaStore Toolbar:

<table>
<thead>
<tr>
<th>Method</th>
<th>File Menu Command</th>
<th>Toolbar Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Dictionary Files manually</td>
<td>From the <em>File</em> menu, use the <em>Add</em> command and select the required data source type from the submenu.</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td><strong>Collecting Dictionary Files</strong></td>
<td><strong>Collecting Dictionary Files</strong>&lt;br&gt;(page 180)&lt;br&gt;From the <em>File</em> menu, use the <em>Collect File</em> command.</td>
<td><img src="image" alt="Icon" /></td>
</tr>
<tr>
<td><strong>Importing MDL Files</strong>&lt;br&gt;(page 241)</td>
<td>From the <em>File</em> menu, use the <em>Import File</em> command.</td>
<td><img src="image" alt="Icon" /></td>
</tr>
</tbody>
</table>
You will build Dictionary Files manually, if it is not possible to collect or import the required metadata. As the procedures for building Dictionary Files are different for different source types, this guide contains separate chapters for each supported source type:

- **Adabas File Group** (page 21)
- **Datacom File Group** (page 46)
- **IMS PCB** (page 71)
- **SQL Table Group** (page 92)
- **Standard File** (page 109)
- **IDMS Subschema** (page 134)
- **Supra Database** (page 159)

Each chapter contains separate procedures for the following phases:

- Creating the Dictionary File
- Defining the Records
- Defining the Fields
- If applicable: Defining Relationships
- If applicable: Defining Indices
This chapter explains the following actions:

- Creating the Dictionary File (page 21)
- Defining Records (page 24)
- Defining Fields (page 28)
- Defining Relationships (page 38)
- Defining Indices (page 42)

For more technical information, refer to the *Adabas File Access Guide*.

### 9.1. Creating the Dictionary File

1. In the Tree View Window, right-click the MetaStore root icon and select Add > Adabas File Group.
   
The properties panel is displayed in the Workspace.

   ![Property Panel](image)

   Two tabs are available: Technical and Business.

2. Fill out the required fields.
   
   For a detailed description of the fields, refer to the sections:
   
   - Technical Tab (Adabas File Groups) (page 22)
   - Business Tab (Adabas Data Objects) (page 45)
3. Apply or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the new Adabas File Group and select Save to MetaStore.

Note: If you do not save the Adabas File Group to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

Technical Tab (Adabas File Groups)

The following fields are available on the Technical tab:

- **Name** (page 22)
- **Code-control** (page 22)
- **Database** (page 23)
- **Version** (page 23)
- **CCSID** (page 23)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Code-control**

Optional field.

Code-control Tables contain information about how to process a certain type of data container. Each table element of the code-control table points to a code table in the COBOL Generator Dictionary.

Each data container type has a pre-defined code-control table. If required, this default code-control table can be overwritten. For example: in order to invoke an external I/O module to read or write the file or to expand or compress data.

Note: When the option is used, the selected table must exist in the COBOL Generator Dictionary before programs using the file can be generated.

For more information about code-control tables, refer to the section Code-control Tables in the Generator Manager Guide.
Database

Enter the name of the corresponding database, if applicable.

Version

Default value = 1
You can change the version number, if required.
When reimporting or recollecting an existing data container, the version will be modified depending on the settings specified in the INI Manager. For more information, refer to the chapter MetaStore Manager Settings in the MetaSuite INI Manager Guide.

CCSID

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.

Business Tab (Adabas File Groups)

The fields on the Business Tab are identical for all Adabas data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Adabas Data Objects) (page 45) for a description of the fields.
9.2. Defining Records

1. In the Tree View Window, right-click the Adabas File Group you want to define Records for and select Add Adabas File.
   
   The properties panel is displayed in the Workspace.

   ![Property Panel](image)

   Two tabs are available: **Technical** and **Business**.

2. Fill out the required fields.
   
   For a detailed description of the fields, refer to the sections:
   
   - Technical Tab (Adabas Files) (page 24)
   - Business Tab (Adabas Data Objects) (page 45)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   
   In the Tree View Window, right-click the Adabas File Group the new Adabas File belongs to and select Save to MetaStore.

   **Note:** If you do not add the File to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Adabas Files)**

The following fields are available on the Technical tab:

- **Name** (page 25)
- **Size** (page 25)
- **Database** (page 25)
- **CCSID** (page 25)
- **Record Keys** (page 26)
**Name**

Mandatory field.
The name you enter must meet the following conditions:
- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Size**

Mandatory field.
Enter the maximum number of characters included in the record. The value must be an integer and cannot exceed the maximum record size defined for the Dictionary File it belongs to.

**Note:** If the size is changed manually and it is too small to contain all defined fields, it will be reset by the MetaStore Manager to the smallest size needed to contain all defined fields.

**Auto Calculate**

If this option is activated, the size will be calculated automatically.

**Database**

**Note:** This field is specific for each file (subtype). Refer to the appropriate File Access Guide for more detailed information.

This field is mandatory for IMS Segments. Enter the segment name of the record, as specified in the DBDGEN and PSBGEN statements.
For RDBMS, the Creator Name should be entered in this field.

**CCSID**

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.
Record Keys

Optional field.
This option is used to identify Adabas File identification fields and the specific ranges of values for those key fields.

1. Double-click the New icon available in the Record Keys panel.

**Note:** You can only define Record Keys after having defined Records and Fields for the Dictionary File.

The following screen is displayed:

2. Fill out the name of the new Key.

3. Select the Key Type from the drop-down list.
   The following options are available:
   - **Access**
   - **Storage**

4. Select the required Field.
   Click the **Browse** button to display the list of all available fields and subfields.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - **Show all**
     When selecting this option, the **Select Item** drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - **Indentation**
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

5. Enter the Value.

**Note:** If you want to specify more than 1 value (or range of values), you have to define separate record keys.
6. **Apply your changes and close the Record Key Properties panel.**
   The new Record Key is added to the Record Keys panel.

7. **Repeat this action for all Record Keys you want to define.**

### Business Tab (Adabas Files)

The fields on the Business Tab are identical for all Adabas data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section  *Business Tab (Adabas Data Objects)* (page 45) for a description of the fields.
9.3. Defining Fields

1. In the Tree View Window, right-click the Adabas File you want to define new Fields for and select Add Adabas Field.

   The properties panel is displayed in the Workspace.

   ![Technical Tab (Adabas Fields)](image)

   Two tabs are available: Technical and Business.

2. Fill out the required field.

   For a detailed description of the fields, refer to the sections:

   - Technical Tab (Adabas Fields) (page 28)
   - Business Tab (Adabas Data Objects) (page 45)

   **Note:** Some of the values can also be modified by double-clicking or selecting them from a drop-down list in the Record Fields Window.

3. Save or discard your changes

4. Save the changes to the MetaStore Repository.

   In the Tree View Window, right-click Adabas File Group the new Field belongs to and select Save to MetaStore.

   **Note:** If you do not add the Field to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Adabas Fields)**

The following fields are available on the Technical tab:
• Properties
  - Name (page 29)
  - Size (page 29)
  - Abs. Position (page 31)
  - Rel. Position (page 31)
  - Occurs (page 31)
  - Occurrence Depending on (page 31)

• Content
  - Type (page 32)
  - Decimals (page 33)
  - Unsigned (page 33)
  - Separated and Leading (page 34)
  - Initial (page 34)
  - Code (page 34)
  - Edit Mask (page 35)
  - Date Format (page 37)
  - Low Limit (page 37)
  - High Limit (page 37)
  - Database (page 38)
  - CCSID (page 38)

**Name**

Mandatory field.
The name you enter must meet the following conditions:

• The name may contain up to 32 characters.
• It must begin with an alphabetic character.
• It may contain the characters a–z, A–Z and 0–9.
• It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
• It may not be ended with a – (hyphen).
• File names must be unique in the MetaStore Repository.

**Size**

In the Size field, enter an integer indicating the field length as a number of bytes. If the field occurs more than once in the record, this field indicates the length of a single occurrence of the field.

<table>
<thead>
<tr>
<th>Type</th>
<th>Additional</th>
<th>Size in # Bytes</th>
<th>COBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td></td>
<td># characters</td>
<td>PIC A(c) DISPLAY</td>
</tr>
<tr>
<td>Type</td>
<td>Additional</td>
<td>Size in # Bytes</td>
<td>COBOL</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>-----------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>Character</td>
<td># characters</td>
<td></td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>National</td>
<td>2 x # characters</td>
<td></td>
<td>PIC N(c)</td>
</tr>
<tr>
<td>Varchar</td>
<td># characters</td>
<td></td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>Binary</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td>Binary Native</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td>Bit</td>
<td>1</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Decimal</td>
<td>Signed / Null-signed</td>
<td>(# digits + 1) / 2</td>
<td>PIC S9(n)V9(d) PACKED-DECIMAL</td>
</tr>
<tr>
<td>Float</td>
<td>8 digits of precision</td>
<td>4</td>
<td>COMP-1</td>
</tr>
<tr>
<td></td>
<td>17 digits of precision</td>
<td>8</td>
<td>COMP-2</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>1 byte = 2 hexadecimal digits</td>
<td></td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>Numeric</td>
<td># digits</td>
<td></td>
<td>PIC S9(n)V9(d) DISPLAY</td>
</tr>
<tr>
<td>Printed Numeric</td>
<td># characters in Edit Mask</td>
<td></td>
<td>PIC EditMask DISPLAY</td>
</tr>
<tr>
<td></td>
<td>2 x # characters in Edit Mask</td>
<td></td>
<td>PIC EditMask USAGE NATIONAL</td>
</tr>
</tbody>
</table>

Legend:
- \(c\) = number of characters
- \(n\) = number of integer digits
- \(d\) = number of decimal digits
- \(s\) = only present when value is signed. This option is not allowed for National.

Note: The Record Layout window shows the default value for the Field within the selected Record. The default value matches the size of the Field, as defined in the Record Fields Window.

Auto Calculate
If this option is activated, the size will be calculated automatically.
Abs. Position

In this field, you may change the Field's starting character position that was calculated automatically by the system. The value entered must be an integer.

Note: If the Record’s length is variable, you may not include the four-character record descriptor word when determining the start position of the Field.

Rel. Position

This is the 1-based position of this field within its domed structure (i.e., Record or Group).

- For a field that is not a subfield, the absolute and relative positions are equal.
- For a field $x$ that is a subfield of group $g$, the relative position will be calculated as $1 + \text{(absolute position of } x) - \text{(absolute position of } g)$. For example: if group $g$ starts on position 5, and the absolute position of $x$ is also 5, then the relative position of $x$ is $1+5-5 = 1$. In human terms: “$x$ will start on the 1st position of $g$”.

Auto Calculate

If this option is activated, the size will be calculated automatically.

Occurs

Optional field. Enter an integer indicating the maximum number of times this field can occur in the record. If you do not define a specific value, the field is assumed to occur once.

Occurrence Depending on

Optional field. Only applies for repeating fields ($n$). Select a numeric field that is defined prior to this field within the record. Results:

- The value you defined with the Occurs option represents the maximum number of times this field can occur.
- The value available in the selected field represents the actual number of times the field occurs.

1. Click the Browse button next to the Occurrence Depending On text field. The list of available numeric fields is displayed.

2. Select the required Field. The name of the selected Field is displayed in the Occurrence Depending On text field.

Note: If you click the Browse button again now, the Properties screen for the selected field will be displayed. If you want to select another field, you first have to delete the current field name from the text field and then click the Browse button.
**Type**

This field is mandatory and contains the default value *Character.*
There are two general classes of data types: *non-numeric* and *numeric.*

### Non-numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td>This datatype indicates a field containing strings using the characters A-Z and a-z.</td>
</tr>
<tr>
<td>Character</td>
<td>This datatype indicates that the field may contain any possible character within the character set being used (including <em>unprintable</em> characters). It is not possible to perform numeric operations on a Character type field, even when the field contains only digits. Matching RDBMS Data Types: CHAR, BYTE, VARCHAR, VARCHAR2, LONG, RAW, LONGVARCHAR</td>
</tr>
<tr>
<td>Graphic</td>
<td>DB2 terminology for Character datatype.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>This datatype indicates that any hexadecimal characters are allowed within the field.</td>
</tr>
<tr>
<td>Long Varchar</td>
<td>This datatype indicates a field containing a variable character string of maximum 2 GB.</td>
</tr>
<tr>
<td>National</td>
<td>The National datatype is a subset of Unicode. The character set on which it is based is UTF-16, but restricted to two bytes per character.</td>
</tr>
<tr>
<td>Varchar</td>
<td>This datatype indicates a character field allowing the storage of values you need plus one byte for the length.</td>
</tr>
<tr>
<td>Vargraphic</td>
<td>DB2 terminology for Varchar datatype.</td>
</tr>
</tbody>
</table>

### Numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>This datatype indicates that the field contains a binary numeric value composed of 0’s and 1’s.</td>
</tr>
<tr>
<td>Binary Native</td>
<td>This datatype indicates that the field contains binary native numbers. This means that its internal representation depends on the Operating System and/or processor.</td>
</tr>
<tr>
<td>Bit</td>
<td>This datatype indicates a field occupying a single bit storage. A BIT type field may only contain the binary values 0 or 1. It is possible to perform numeric operations on a BIT type field.</td>
</tr>
<tr>
<td>Byte</td>
<td>This datatype indicates a field occupying a single byte storage. A byte is 8 bits. It is possible to perform numeric operations on a BYTE type field.</td>
</tr>
<tr>
<td>Decimal</td>
<td>This datatype indicates a field containing a packed decimal value. Packed decimal is the most commonly used internal numeric data type. Two decimal digits are contained in each byte of a packed decimal number. However, if the number is signed, the last half byte contains a positive or negative sign indicator.</td>
</tr>
</tbody>
</table>
Decimals

If the field type allows the definition of decimals, this field contains the number of decimals. Decimals can be entered for the following field types:

- Binary
- Binary Native
- Decimal
- Numeric
- Printed Numeric
- Printed Numeric National

If the field type does not allow the definition of decimals, this field contains the default value 0.

Unsigned

Select the Unsigned check box to indicate that a numeric value is not signed, i.e. that it does not contain a sign indicator (+ or -).

This field applies for the following field types:

- Binary
- Binary Native
- Numeric

Note: For Decimal fields, the distinction between unsigned and positive signed INPUT fields is not relevant to the system.
Separated and Leading

These two check boxes allow to define how over-punched and separate sign indicators must be treated. They only apply for the Numeric field type.

There are several possibilities.

If the sign indicator is over-punched in the number, select the *Unsigned* check box above.

- Select *Leading* to put the sign on the initial digit.
- Select *Separate* to put the sign on the last byte (no digit).
- Select *Leading* and *Separate* to put the sign in the initial byte (no digit).

If the Numeric field contains a separate plus or minus sign attached to the number, select the *Separate* check box.

Initial

Optional field.

Enter the initial value for the field.

This value is only taken into account for Target Files. If an initial value was defined for a Source file, this parameter will not affect any operation, except for syntax checking.

For Numeric fields, this is a numeric constant, where the decimal point should be given by a . (point).

For Character fields, this may be an alphanumeric constant (not enclosed by quotes) or the system fields SYS-LOW-VALUE or SYS-HIGH-VALUE, which correspond to system fields LOW-VALUE and HIGH-VALUE within COBOL.

Code

Default value = *No Code*

The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>This field is treated as a code, not as a number.</td>
</tr>
<tr>
<td>No Code</td>
<td>Resets the code operator to zero. Select this option when there is no additional information to be added for the field.</td>
</tr>
<tr>
<td>Time</td>
<td>Select this option to define a field that contains TIME information.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Select this option to define a field that contains TIMESTAMP information.</td>
</tr>
</tbody>
</table>
Edit Mask

This field is mandatory for Printed Numeric fields, but optional for other field types. It indicates how the alphanumeric values must be formatted.

**Note:** When the Edit Mask is set for a printed numeric field, the MetaStore Manager will reset the size of the field to the size that corresponds to the chosen Edit Mask.

Enter the characters defining a Mask in this text field. This Mask will override the default mask for the field. There is a default mask for each field type. Both the default masks and the manually created masks are composed of *Replacement* and *Insertion* characters.

The following table lists the Replacement characters and their meaning. Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field.

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>N</td>
<td>National (2-byte Unicode character set)</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
</tbody>
</table>

The following table lists the Insertion characters and their meaning. Insertion characters indicate characters to be printed in addition to those contained in the stored field.

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
<tr>
<td>,</td>
<td>Comma (separator for the sake of large number readability or decimal separator depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>.</td>
<td>Decimal point (separator for the sake of large number readability or decimal separator, depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>B</td>
<td>Blank</td>
</tr>
<tr>
<td>-</td>
<td>Floating or trailing minus for negative values. Plus is not accepted as valid character. Positive numbers will have a blank as sign character.</td>
</tr>
<tr>
<td>+</td>
<td>Floating or trailing plus or minus sign</td>
</tr>
<tr>
<td>CR</td>
<td>Trailing credit symbol for negative values only</td>
</tr>
</tbody>
</table>
As mentioned above, there is a default Mask for each field type:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Default Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields</td>
<td>The default mask contains a minus sign as the rightmost character. All negative values are printed with a trailing minus sign.</td>
</tr>
<tr>
<td>Numeric fields with decimals</td>
<td>The default mask contains a decimal point and as many digit replacement characters (9s) to the right of the decimal point as are specified by the Decimal option.</td>
</tr>
<tr>
<td>Numeric fields</td>
<td>The default mask contains as many digits as its size, without zero suppression.</td>
</tr>
<tr>
<td>Date fields</td>
<td>The default mask is its selected date format.</td>
</tr>
<tr>
<td>Alphanumeric fields</td>
<td>The default mask contains as many alphanumeric character replacement characters (X) as are required to print the field.</td>
</tr>
</tbody>
</table>

Examples of default masks:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Size</th>
<th>Default mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields without decimal positions</td>
<td>6</td>
<td>999999-</td>
</tr>
<tr>
<td>Signed numeric fields with two decimals</td>
<td>6</td>
<td>9999.99-</td>
</tr>
<tr>
<td>Character Field</td>
<td>6</td>
<td>X(6)</td>
</tr>
<tr>
<td>Unicode Field</td>
<td>6</td>
<td>N(3)</td>
</tr>
</tbody>
</table>

You may also define customized masks.
Examples:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>6</td>
<td>XXBXXX</td>
<td>AB138B</td>
<td>AB 138B</td>
</tr>
<tr>
<td>Numeric with 2 decimal positions</td>
<td>2</td>
<td>.99</td>
<td>.35</td>
<td>.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-.12</td>
<td>-.12</td>
</tr>
<tr>
<td>Numeric with 3 minus signs</td>
<td>4</td>
<td>...-9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>
Date Format

If the field must contain a date, select the required date format from the drop-down list. The system automatically validates date fields whenever they are referenced in a MetaMap model, and automatically converts date fields whenever they are compared to another date or used in a calculation. In all the available formats, YY or YYYY stands for the year and MM stands for the month. DD stands for the day within a month and DDD stands for the day within the year.

When the format contains a '?', this indicates the date delimiter that is used. Data formats with a '?' are only supported for Character and Varchar field types. When the data format does not contain a '?', the different parts in the date are not delimited by a special character.

The Date Format list is accessible for the following field types:

- Binary
- Binary Native
- Character
- Decimal
- Numeric
- National
- Varchar

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric with 3 plus signs</td>
<td>4</td>
<td>+++9</td>
<td>0</td>
<td>+0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>

Note: When a date format is chosen, MetaStore Manager will reset the size of the field to the size that corresponds to the chosen date format.

Low Limit

Optional field.

When a Low Limit is specified, you must define a value in the High Limit field as well.

You may use this field to define a minimum value for this field. If a lower value is encountered, the system will consider the data invalid.

High Limit

Optional field.

When a High Limit is specified, you must define a value in the Low Limit field as well. You may use this field to define a maximum value for this field. If a higher value is encountered, the system will consider the data invalid.
Database

Enter the name of the corresponding database, if applicable.

CCSID

Enter the Coded Character Set Identifier.

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.

This CCSID is an enriched property and will not be collected.

Business Tab (Adabas Fields)

The fields on the Business Tab are identical for all Adabas data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Adabas Data Objects) (page 45) for a description of the fields.

9.4. Defining Relationships

1. In the Tree View Window, right-click the Adabas File Group you want to define a Relationship for and select Add Adabas Relationship.

   The properties panel is displayed in the Workspace.

   ![Image of the properties panel]

   Each LINK definition relates two MetaSuite (ADABAS/C) records to one another. The "basis" of the link is a field in one record (the "from" record, so called because data is taken from that record and used to locate the other record) and a descriptor in the other (the "to" record, so called because the descriptor leads to that record).

   Two tabs are available: Technical and Business.
2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Adabas Relationships) (page 39)
   - Business Tab (Adabas Data Objects) (page 45)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Adabas File Group the new Relationship belongs to and select "Save to MetaStore."

   **Note:** If you do not add the Relationship to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Adabas Relationships)**

The following fields are available on the Technical tab:

- **Name** (page 39)
- **Database** (page 39)
- **Relationship From** (page 40)
- **Relationships To** (page 41)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Database**

This is the unique name for the Database Entity (32 characters).
Relationship From

Mandatory field.

1. Fill out the name of the new Relationship key.

2. Click the **Browse** button next to the **Relationship From** text field.
   
   The following screen is displayed:

   ![Relationship From Screen](image)

3. Select the required File Group or File from the drop-down list and click **OK**.
   
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   
   - **Show all**
     
     When selecting this option, the **Select Item** drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   
   - **Indentation**
     
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

   The **Adabas Relationship From** properties panel is displayed.

4. Fill out the required fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>The name of the new Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Record</td>
<td>In order to define a particular relationship between two records, the user has to specify the first record of the relationship in the <strong>From Record</strong> field.</td>
</tr>
<tr>
<td>From Field</td>
<td>The matching field of the <strong>From Record</strong>.</td>
</tr>
</tbody>
</table>

**Note:** You can use the **Browse** button to display the list of available items.
5. Apply your changes.
   The name will be displayed in the Relationship From field.

**Relationships To**

1. Double-click the New icon available in the Relationships To text zone.
   The following screen is displayed:

2. Select the required File Group or File. Two extra options are available at the top right of the pop-up window for selecting the required item:
   - **Show all**
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - **Indentation**
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

3. Select the required Field and click OK.
   The Adabas Relationship To properties panel is displayed.
4. Fill out the required fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the new Relationship</td>
</tr>
<tr>
<td>To Record</td>
<td>In order to define a particular relationship between two records, the user has to specify the second record of the relationship in the To Record field.</td>
</tr>
<tr>
<td>To Field</td>
<td>The matching field of the To Record.</td>
</tr>
<tr>
<td>To Index</td>
<td>For faster access, the user can define an index on a table. This index has to be specified in the MetaStore database and can be referred to in this field.</td>
</tr>
</tbody>
</table>

**Note:** You can use the Browse button to display the list of available items.

5. Apply your changes.

The relationship key name is added in the Relationships To panel.

**Business Tab (Adabas Relationships)**

The fields on the Business Tab are identical for all Adabas data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Adabas Data Objects) (page 45) for a description of the fields.

### 9.5. Defining Indices

1. In the Tree View Window, right-click the Adabas File Group you want to define an Index for and select Add Adabas Index.

   The properties panel is displayed in the Workspace.

   ![Index properties panel]

   Two tabs are available: Technical and Business.

2. Fill out the required fields.
For a detailed description of the fields, refer to the sections:

- Technical Tab (Adabas Indices) (page 43)
- Business Tab (Adabas Data Objects) (page 45)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Adabas File Group the new Index belongs to and select Save to MetaStore.

**Note:** If you do not add the Index to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Adabas Indices)**

The following fields are available on the Technical tab:

- Name (page 43)
- Database (page 43)
- Type (page 43)
- Size (page 44)
- Based on Record (page 44)
- Index Keys (page 44)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).

**Database**

Enter the name of the corresponding database.

**Type**

Select the required type from the drop-down list.

The following options are available:

- Binary
• Mixed
• Packed
• Packed Unsigned
• Zoned
• Zoned Unsigned

Size

Enter the size of the index key.

Based on Record

This field is not used for Adabas File Groups.

Index Keys

Mandatory field.

Note: You can only define Index Keys after having saved the Index.

1. Double-click the New icon available in the Index Keys panel.
   The following screen is displayed.

2. Select the required Field and click OK.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   • Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   • Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.
The **Adabas Index Key** properties panel is displayed.

![Adabas Index Key properties panel](image)

3. Fill out the required fields.

   - **Name**: The name of the new Relationship
   - **Field**: Name of the field on which the index key is applied
   - **Begin Position**: This field indicates the begin position of the field.
   - **End Position**: This field indicates the end position of the field.

4. Apply your changes.
   The Index Key name is added in the **Index Keys** panel.

**Business Tab (Adabas Indices)**

The fields on the Business Tab are identical for all Adabas data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section **Business Tab (Adabas Data Objects)** (page 45) for a description of the fields.

9.6. **Business Tab (Adabas Data Objects)**

The following fields are available on the Business tab:

- **Business Rule** (page 45)
- **Note** (page 45)

**Note**: If you want to enter text in RTF (Rich Text Format), right-click and select **RTF** from the context menu (or use the shortcut **CTRL + R**).

**Business Rule**

Optional field.
Enter a free-form description for the data object (Dictionary File, Record, Field, Relationship or Index). For example, the business rule or the requirements.

**Note**

Optional field.
Enter a free-form note for the data object.
CHAPTER 10

Datacom File Group

This chapter explains the following actions:

- **Creating the Dictionary File** (page 46)
- **Defining Records** (page 49)
- **Defining Fields** (page 52)
- **Defining Relationships** (page 63)
- **Defining Indices** (page 67)

10.1. Creating the Dictionary File

1. In the Tree View Window, right-click the MetaStore root icon and select Add > Datacom File Group.

   The properties panel is displayed in the Workspace.

   ![New Datacom File Group](image)

   Two tabs are available: **Technical** and **Business**.

2. Fill out the required fields.

   For a detailed description of the fields, refer to the sections:
   - **Technical Tab (Datacom File Groups)** (page 47)
   - **Business Tab (Datacom Data Objects)** (page 69)
3. Apply or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the new Datacom File Group and select **Save to MetaStore**.

**Note:** If you do not save the Datacom File Group to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

### Technical Tab (Datacom File Groups)

The following fields are available on the Technical tab:

- **Name** (page 47)
- **Code-control** (page 47)
- **Database** (page 48)
- **Version** (page 48)
- **CCSID** (page 48)

### Name

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

### Code-control

Optional field.

Code-control Tables contain information about how to process a certain type of data container. Each table element of the code-control table points to a code table in the COBOL Generator Dictionary.

Each data container type has a pre-defined code-control table. If required, this default code-control table can be overwritten. For example: in order to invoke an external I/O module to read or write the file or to expand or compress data.

**Note:** When the option is used, the selected table must exist in the COBOL Generator Dictionary before programs using the file can be generated.

For more information about code-control tables, refer to the section **Code-control Tables** in the **Generator Manager Guide**.
**Database**

Enter the name of the corresponding database, if applicable.

**Version**

Default value = 1  
You can change the version number, if required.  
When reimporting or recollecting an existing data container, the version will be modified depending on the settings specified in the INI Manager. For more information, refer to the chapter MetaStore Manager Settings in the *MetaSuite INI Manager Guide*.

**CCSID**

Enter the Coded Character Set Identifier.  
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.  
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.  
This CCSID is an enriched property and will not be collected.

**Business Tab (Datacom File Groups)**

The fields on the Business Tab are identical for all Datacom data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Datacom Data Objects) (page 69) for a description of the fields.
10.2. Defining Records

1. In the Tree View Window, right-click the Datacom File Group you want to define Records for and select Add Datacom Record.

   The properties panel is displayed in the Workspace.

   ![Properties Panel](image)

   This screen contains two tabs: Technical and Business.

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Datacom Records) (page 49)
   - Business Tab (Datacom Data Objects) (page 69)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Datacom File Group the new Record belongs to and select Save to MetaStore.

   **Note:** If you do not add the Record to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Datacom Records)**

The following fields are available on the Technical tab:

- Name (page 50)
- Size (page 50)
- Database (page 50)
- CCSID (page 50)
- Record Keys (page 51)
Name

Mandatory field.
The name you enter must meet the following conditions:
- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

Size

Mandatory field.
Enter the maximum number of characters included in the record. The value must be an integer and cannot exceed the maximum record size defined for the Dictionary File it belongs to.

Note: If the size is changed manually and it is too small to contain all defined fields, it will be reset by the MetaStore Manager to the smallest size needed to contain all defined fields.

Auto Calculate
If this option is activated, the size will be calculated automatically.

Database

Note: This field is specific for each file (sub)type. Refer to the appropriate File Access Guide for more detailed information.

This field is mandatory for IMS Segments. Enter the segment name of the record, as specified in the DBDGEN and PSBGEN statements.
For RDBMS, the Creator Name should be entered in this field.

CCSID

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.
Record Keys

Optional field.
This option is used to identify Datacom identification fields and the specific ranges of values for those key fields.

1. Double-click the New icon available in the Record Keys panel.

   **Note:** You can only define Record Keys after having defined Records and Fields for the Dictionary File.

   The following screen is displayed:

   ![Record Keys Screen](image)

2. Fill out the name of the new Key.

3. Select the Key Type from the drop-down list.
   The following options are available:
   - **Access:** this is the Master Key, used to determine the read order
   - **Storage:** this is the Native Key, used to determine the write or store order

4. Select the required Field
   Click the *Browse* button to display the list of all available fields and subfields.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - **Show all**
     When selecting this option, the *Select Item* drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - **Indentation**
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

5. Select the required Index
   Click the *Browse* button to display the list of all available fields and subfields.

6. Enter the Value.

   **Note:** If you want to specify more than 1 value (or range of values), you have to define separate record keys.

   Mandatory field.
Only applicable when *Access* is selected as Key Type. This value defines the value of the field indicating an occurrence of the record.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Value entered</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal</td>
<td>3</td>
<td>Only records where the selected field has (not) value 3 are taken into account.</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Only records where the selected field has (not) value C are taken into account.</td>
</tr>
</tbody>
</table>

7. Apply your changes and close the *Record Key Properties* panel. The new Record Key is added to the *Record Keys* panel.

8. Repeat this action for all Record Keys you want to define.

**Business Tab (Datacom Records)**

The fields on the Business Tab are identical for all Datacom data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section *Business Tab (Datacom Data Objects)* (page 69) for a description of the fields.

### 10.3. Defining Fields

1. In the Tree View Window, right-click the Datacom Record you want to define new Fields for and select *Add Datacom Field*.

   The properties panel is displayed in the Workspace.

   Two tabs are available: *Technical* and *Business*. 

---

**IKAN Solutions**

**Datacom File Group**

**MetaSuite MetaStore Manager - Release 8.1.3**
2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Datacom Fields) (page 53)
   - Business Tab (Datacom Data Objects) (page 69)

   **Note:** Some of the values can also be modified by double-clicking or selecting them from a drop-down list in the Record Fields Window.

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Datacom File Group the new Field belongs to and select **Save to MetaStore**.

   **Note:** If you do not add the Field to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

### Technical Tab (Datacom Fields)

The following fields are available on the Technical tab:

- **Properties**
  - Name (page 54)
  - Size (page 54)
  - Abs. Position (page 55)
  - Rel. Position (page 55)
  - Occurs (page 55)
  - Occurrence Depending On (page 56)

- **Content**
  - Type (page 56)
  - Decimals (page 57)
  - Unsigned (page 58)
  - Separated and Leading (page 58)
  - Initial (page 58)
  - Null (page 58)
  - Code (page 59)
  - Edit Mask (page 60)
  - Date Format (page 62)
  - Low Limit (page 62)
  - High Limit (page 62)
  - CCSID (page 63)
Name

Mandatory field.
The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

Size

In the Size field, enter an integer indicating the field length as a number of bytes. If the field occurs more than once in the record, this field indicates the length of a single occurrence of the field.

<table>
<thead>
<tr>
<th>Type</th>
<th>Additional</th>
<th>Size in # Bytes</th>
<th>COBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td># characters</td>
<td>PIC A(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Character</td>
<td># characters</td>
<td>PIC X(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>National</td>
<td>2 x # characters</td>
<td>PIC N(c)</td>
<td></td>
</tr>
<tr>
<td>Varchar</td>
<td># characters</td>
<td>PIC X(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Binary</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td>Binary Native</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td>Bit</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decimal</td>
<td>Signed / Null-signed</td>
<td>(# digits + 1) / 2</td>
<td>PIC S9(n)V9(d) PACKED-DECIMAL</td>
</tr>
<tr>
<td>Float</td>
<td>8 digits of precision</td>
<td>4</td>
<td>COMP-1</td>
</tr>
<tr>
<td></td>
<td>17 digits of precision</td>
<td>8</td>
<td>COMP-2</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>1 byte = 2 hexadecimal digits</td>
<td>PIC X(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Numeric</td>
<td># digits</td>
<td>PIC S9(n)V9(d) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Printed Numeric</td>
<td># characters in Edit Mask</td>
<td>PIC EditMask DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Printed Numeric</td>
<td>2 x # characters in Edit Mask</td>
<td>PIC EditMask USAGE NATIONAL</td>
<td></td>
</tr>
</tbody>
</table>
Legend:
- \( c \) = number of characters
- \( n \) = number of integer digits
- \( d \) = number of decimal digits
- \( s \) = only present when value is signed. This option is not allowed for National.

**Note:** The Record Layout window shows the default value for the Field within the selected Record. The default value matches the size of the Field, as defined in the Record Fields Window.

**Auto Calculate**
If this option is activated, the size will be calculated automatically.

**Abs. Position**
In this field, you may change the Field's starting character position that was calculated automatically by the system. The value entered must be an integer.

**Note:** If the Record’s length is variable, you may not include the four-character record descriptor word when determining the start position of the Field.

**Rel. Position**
This is the 1-based position of this field within its domed structure (i.e., Record or Group).
- For a field that is not a subfield, the absolute and relative positions are equal.
- For a field \( x \) that is a subfield of group \( g \), the relative position will be calculated as \( 1 + \) (absolute position of \( x \)) - (absolute position of \( g \)).
  For example: if group \( g \) starts on position 5, and the absolute position of \( x \) is also 5, then the relative position of \( x \) is \( 1+5-5 = 1 \). In human terms: “\( x \) will start on the 1st position of \( g \)”.

**Auto Calculate**
If this option is activated, the size will be calculated automatically.

**Occurs**
Optional field.
Enter an integer indicating the maximum number of times this field can occur in the record. If you do not define a specific value, the field is assumed to occur once.
Occurrence Depending On

Optional field. Only applies for repeating fields (n).
Select a numeric field that is defined prior to this field within the record.
Results:
- The value you defined with the Occurs option represents the maximum number of times this field can occur
- The value available in the selected field represents the actual number of times the field occurs.

1. Click the Browse button next to the Occurrence Depending On text field.
The list of available numeric fields is displayed.

2. Select the required Field.
The name of the selected Field is displayed in the Occurrence Depending On text field.

Note: If you click the Browse button again now, the Properties screen for the selected field will be displayed. If you want to select another field, you first have to delete the current field name from the text field and then click the Browse button.

Type

This field is mandatory and contains the default value Character.
There are two general classes of data types: non-numeric and numeric.

Non-numeric Data Types

<table>
<thead>
<tr>
<th>Alphabetic</th>
<th>This datatype indicates a field containing strings using the characters A-Z and a-z.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>This datatype indicates that the field may contain any possible character within the character set being used (including unprintable characters). It is not possible to perform numeric operations on a Character type field, even when the field contains only digits. Matching RDBMS Data Types: CHAR, BYTE, VARCHAR, VARCHAR2, LONG, RAW, LONGVARCHAR</td>
</tr>
<tr>
<td>Graphic</td>
<td>DB2 terminology for Character datatype.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>This datatype indicates that any hexadecimal characters are allowed within the field.</td>
</tr>
<tr>
<td>Long Varchar</td>
<td>This datatype indicates a field containing a variable character string of maximum 2 GB.</td>
</tr>
<tr>
<td>National</td>
<td>The National datatype is a subset of Unicode. The character set on which it is based is UTF-16, but restricted to two bytes per character.</td>
</tr>
<tr>
<td>Varchar</td>
<td>This datatype indicates a character field allowing the storage of values you need plus one byte for the length.</td>
</tr>
<tr>
<td>Vargraphic</td>
<td>DB2 terminology for Varchar datatype.</td>
</tr>
</tbody>
</table>
# Numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>This datatype indicates that the field contains a binary numeric value composed of 0's and 1's.</td>
</tr>
<tr>
<td>Binary Native</td>
<td>This datatype indicates that the field contains binary native numbers. This means that its internal representation depends on the Operating System and/or processor.</td>
</tr>
<tr>
<td>Bit</td>
<td>This datatype indicates a field occupying a single bit storage. A BIT type field may only contain the binary values 0 or 1. It is possible to perform numeric operations on a BIT type field.</td>
</tr>
<tr>
<td>Byte</td>
<td>This datatype indicates a field occupying a single byte storage. A byte is 8 bits. It is possible to perform numeric operations on a BYTE type field.</td>
</tr>
<tr>
<td>Decimal</td>
<td>This datatype indicates a field containing a packed decimal value. Packed decimal is the most commonly used internal numeric data type. Two decimal digits are contained in each byte of a packed decimal number. However, if the number is signed, the last half byte contains a positive or negative sign indicator.</td>
</tr>
<tr>
<td>Float</td>
<td>This datatype indicates that the field contains floating-point numbers, encoded in an encoded exponential form.</td>
</tr>
<tr>
<td>Numeric</td>
<td>This datatype indicates that any the field contains decimal numbers in a printable character format, this means that a single digit is stored per byte. Leading blanks are not permitted in this field type, nor may it contain editing characters, such as commas or decimal points. Use Printed Numeric in this case. Matching RDBMS Data Types: TINYINT, SMALLINT, INTEGER, BIGINT, DECIMAL, NUMBER, FLOAT</td>
</tr>
<tr>
<td>Printed Numeric</td>
<td>This datatype indicates that the character-based field contains a numeric value. The format in which the printed numeric value is displayed must be specified with the EDIT option.</td>
</tr>
<tr>
<td>Printed Numeric National</td>
<td>This datatype indicates that this is a National field that contains a numeric value. The format in which the PRN-NATIONAL value is displayed must be specified with the EDIT option.</td>
</tr>
</tbody>
</table>

## Decimals

If the field type allows the definition of decimals, this field contains the number of decimals. Decimals can be entered for the following field types:

- Binary
- Binary Native
- Decimal
- Numeric
- Printed Numeric
- Printed Numeric National

If the field type does not allow the definition of decimals, this field contains the default value 0.
Unsigned

Select the **Unsigned** check box to indicate that a numeric value is not signed, i.e. that it does not contain a sign indicator (+ or -).

This field applies for the following field types:

- Binary
- Binary Native
- Numeric

**Note:** For Decimal fields, the distinction between unsigned and positive signed INPUT fields is not relevant to the system.

Separated and Leading

These two check boxes allow to define how over-punched and separate sign indicators must be treated. They only apply for the Numeric field type.

There are several possibilities.

If the sign indicator is over-punched in the number, select the **Unsigned** check box above.

- Select **Leading** to put the sign on the initial digit.
- Select **Separate** to put the sign on the last byte (no digit).
- Select **Leading** and **Separate** to put the sign in the initial byte (no digit).

If the Numeric field contains a separate plus or minus sign attached to the number, select the **Separate** check box.

Initial

Optional field.

Enter the initial value for the field.

This value is only taken into account for Target Files. If an initial value was defined for a Source file, this parameter will not affect any operation, except for syntax checking.

For Numeric fields, this is a numeric constant, where the decimal point should be given by a . (point).

For Character fields, this may be an alphanumeric constant (not enclosed by quotes) or the system fields SYS-LOW-VALUE or SYS-HIGH-VALUE, which correspond to system fields LOW-VALUE and HIGH-VALUE within COBOL.

Null

This field indicates the nullable status of the field, and whether Inbound or Outbound nulls are used.

**Note:** If this field is left blank, the default NULLABLE value defined in the MetaSuite dictionary will be applied to this field. For more information, refer to the section Create Dictionary/Enter License Key in the Generator Manager Guide.

Select the required null-indicator from the drop-down list.
The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>This field is treated as a code, not as a number.</td>
</tr>
<tr>
<td>No Code</td>
<td>Resets the code operator to zero. Select this option when there is no additional information to be added for the field.</td>
</tr>
<tr>
<td>Time</td>
<td>Select this option to define a field that contains TIME information.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Select this option to define a field that contains TIMESTAMP information.</td>
</tr>
</tbody>
</table>
Edit Mask

This field is mandatory for Printed Numeric fields, but optional for other field types. It indicates how the alphanumeric values must be formatted.

**Note:** When the Edit Mask is set for a printed numeric field, the MetaStore Manager will reset the size of the field to the size that corresponds to the chosen Edit Mask.

Enter the characters defining a Mask in this text field. This Mask will override the default mask for the field. There is a default mask for each field type. Both the default masks and the manually created masks are composed of Replacement and Insertion characters.

The following table lists the Replacement characters and their meaning. Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field.

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>N</td>
<td>National (2-byte Unicode character set)</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
</tbody>
</table>

The following table lists the Insertion characters and their meaning. Insertion characters indicate characters to be printed in addition to those contained in the stored field.

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
<tr>
<td>,</td>
<td>Comma (separator for the sake of large number readability or decimal separator depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>.</td>
<td>Decimal point (separator for the sake of large number readability or decimal separator, depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>B</td>
<td>Blank</td>
</tr>
<tr>
<td>-</td>
<td>Floating or trailing minus for negative values. Plus is not accepted as valid character. Positive numbers will have a blank as sign character.</td>
</tr>
<tr>
<td>+</td>
<td>Floating or trailing plus or minus sign</td>
</tr>
<tr>
<td>CR</td>
<td>Trailing credit symbol for negative values only</td>
</tr>
</tbody>
</table>
As mentioned above, there is a default mask for each field type:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Default Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields</td>
<td>The default mask contains a minus sign as the rightmost character. All negative values are printed with a trailing minus sign.</td>
</tr>
<tr>
<td>Numeric fields with decimals</td>
<td>The default mask contains a decimal point and as many digit replacement characters (9s) to the right of the decimal point as are specified by the Decimal option.</td>
</tr>
<tr>
<td>Numeric fields</td>
<td>The default mask contains as many digits as its size, without zero suppression.</td>
</tr>
<tr>
<td>Date fields</td>
<td>The default mask is its selected date format.</td>
</tr>
<tr>
<td>Alphanumeric fields</td>
<td>The default mask contains as many alphanumeric character replacement characters (X) as are required to print the field.</td>
</tr>
</tbody>
</table>

Examples of default masks:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Size</th>
<th>Default mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields without decimal positions</td>
<td>6</td>
<td>999999-</td>
</tr>
<tr>
<td>Signed numeric fields with two decimals</td>
<td>6</td>
<td>9999.99-</td>
</tr>
<tr>
<td>Character Field</td>
<td>6</td>
<td>X(6)</td>
</tr>
<tr>
<td>Unicode Field</td>
<td>6</td>
<td>N(3)</td>
</tr>
</tbody>
</table>

You may also define customized masks. Examples:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>6</td>
<td>XXBXXX</td>
<td>AB138B</td>
<td>AB 138B</td>
</tr>
<tr>
<td>Numeric with 2 decimal positions</td>
<td>2</td>
<td>.99</td>
<td>.35</td>
<td>.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-.12</td>
<td>.12</td>
</tr>
<tr>
<td>Numeric with 3 minus signs</td>
<td>4</td>
<td>---9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>
**Date Format**

If the field must contain a date, select the required date format from the drop-down list. The system automatically validates date fields whenever they are referenced in a MetaMap model, and automatically converts date fields whenever they are compared to another date or used in a calculation. In all the available formats, YY or YYYY stands for the year and MM stands for the month. DD stands for the day within a month and DDD stands for the day within the year.

When the format contains a '?', this indicates the date delimiter that is used. Data formats with a '?' are only supported for Character and Varchar field types. When the data format does not contain a '?', the different parts in the date are not delimited by a special character.

The Date Format list is accessible for the following field types:

- Binary
- Binary Native
- Character
- Decimal
- Numeric
- National
- Varchar

**Note:** When a date format is chosen, MetaStore Manager will reset the size of the field to the size that corresponds to the chosen date format.

**Low Limit**

Optional field.

When a *Low Limit* is specified, you must define a value in the *High Limit* field as well. You may use this field to define a minimum value for this field. If a lower value is encountered, the system will consider the data invalid.

**High Limit**

Optional field.

When a *High Limit* is specified, you must define a value in the *Low Limit* field as well. You may use this field to define a maximum value for this field. If a higher value is encountered, the system will consider the data invalid.

---

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric with 3 plus signs</td>
<td>4</td>
<td>+++9</td>
<td>0</td>
<td>+0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>210</td>
<td>210</td>
<td></td>
</tr>
</tbody>
</table>
CCSID

Enter the Coded Character Set Identifier. CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page. CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL. This CCSID is an enriched property and will not be collected.

Business Tab (Datacom Fields)

The fields on the Business Tab are identical for all Datacom data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Datacom Data Objects) (page 69) for a description of the fields.

10.4. Defining Relationships

1. In the Tree View Window, right-click the Datacom File Group you want to define a Relationship for and select Add Datacom Relationship. The properties panel is displayed in the Workspace.

   Each LINK definition relates two MetaSuite (Datacom File Group) records to one another. The "basis" of the link is a field in one record (the "from" record, so called because data is taken from that record and used to locate the other record) and a descriptor in the other (the "to" record, so called because the descriptor leads to that record). Two tabs are available: Technical and Business.

2. Fill out the required fields. For a detailed description of the fields, refer to the sections:
   - Technical Tab (Datacom Relationships) (page 64)
   - Business Tab (Datacom Data Objects) (page 69)
3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Datacom File Group and select *Save to MetaStore.*

**Note:** If you do not save the Relationship to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Datacom Relationships)**

The following fields are available on the Technical tab:

- **Name** (page 64)
- **Relationship From** (page 64)
- **Relationships To** (page 66)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Relationship From**

1. Fill out the name of the new Relationship key.

2. Click the *Browse* button next to the *Relationship From* text field.
   A screen similar to this one is displayed:
3. First select the required Item (File Group or Record) using the drop-down list. All available fields for the selected record or fields will be displayed underneath. Two extra options are available at the top right of the pop-up window for selecting the required item:
   • Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   • Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

4. Select the required record or field and click OK.
   The Datacom Relationship From properties panel is displayed.

   ![New Datacom Relationship From properties panel](image)

5. Fill out the required fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the new Relationship</td>
</tr>
<tr>
<td>From Record</td>
<td>In order to define a particular relationship between two records, the user has to specify the first record of the relationship in the From Record field.</td>
</tr>
<tr>
<td>From Field</td>
<td>The matching field of the From Record.</td>
</tr>
</tbody>
</table>

   **Note:** You can use the Browse button to display the list of available items.

6. Apply your changes and close the properties panel.
   The name will be displayed in the Relationship From field.
Relationships To

1. Double-click the New icon available in the Relationships To text zone.
   A screen similar to this one is displayed:

   ![Image of the Relationships To screen]

2. First select the required Record using the drop-down list.
   All available fields for the selected record will be displayed underneath.

3. Select the required Field and click OK.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields
     for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.
   The Datacom Relationship To properties panel is displayed.

   ![Image of the Datacom Relationship To properties panel]
4. Fill out the required fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the new Relationship</td>
</tr>
<tr>
<td>To Record</td>
<td>In order to define a particular relationship between two records, the user</td>
</tr>
<tr>
<td></td>
<td>has to specify the second record of the relationship in the To Record field.</td>
</tr>
<tr>
<td>To Field</td>
<td>The matching field of the To Record.</td>
</tr>
<tr>
<td>To Index</td>
<td>The matching index of the To Record.</td>
</tr>
</tbody>
</table>

**Note:** You can use the Browse button to display the list of available items.

5. Apply your changes and close the properties panel.

   The relationship key name is added in the Relationships To panel.

**Business Tab (Datacom Relationships)**

The fields on the Business Tab are identical for all Datacom data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Datacom Data Objects) (page 69) for a description of the fields.

**10.5. Defining Indices**

1. In the Tree View Window, right-click the Datacom File Group you want to define an Index for and select Add Datacom Index.

   The properties panel is displayed in the Workspace:

   ![Image of the properties panel](image)

   Two tabs are available: **Technical** and **Business**.

2. Fill out the required fields.

   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Datacom Indices) (page 68)
   - Business Tab (Datacom Data Objects) (page 69)
3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Datacom File Group the new Index belongs to and select Save to MetaStore.

   **Note:** If you do not add the Index to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Datacom Indices)**
The following fields are available on the Technical tab:

- **Name** (page 68)
- **Database** (page 68)
- **Size** (page 68)
- **Based on Record** (page 69)

**Name**
Mandatory field.
The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a–z, A–Z and 0–9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).

**Database**
Enter the name of the corresponding database. This name is limited to 5 characters.

**Size**
Enter the size of the index key.
Based on Record

1. Click the Browse button next to the Based on Record text field to select the Record. A screen similar to this one is displayed:

   ![Image of Browse screen]

2. First select the required Item (File Group or Record) using the drop-down list. All available fields for the selected record or fields will be displayed underneath. Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

3. Select the required Record and click OK. The name of the Record is now entered in the Based on Record field.

Business Tab (Datacom Indices)

The fields on the Business Tab are identical for all Datacom data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Datacom Data Objects) (page 69) for a description of the fields.

10.6. Business Tab (Datacom Data Objects)

The following fields are available on the Business tab:
- Business Rule (page 70)
- Note (page 70)

Note: If you want to enter text in RTF (Rich Text Format), right-click and select RTF from the context menu (or use the shortcut CTRL + R).
**Business Rule**
Optional field.
Enter a free-form description for the data object (Dictionary File, Record, Field, Relationship or Index). For example, the business rule or the requirements.

**Note**
Optional field.
Enter a free-form note for the data object.
IMS is a database management system (DBMS) for z/OS environments. An IMS application of a database is called a PCB (Program Control Block). It is treated as a single MetaSuite Dictionary File.

In the remainder of this chapter, a Dictionary File for an IMS PCB is referred to as an IMS PCB Dictionary File. The following table shows how the IMS PCB terminology can be matched with the MetaSuite terminology:

<table>
<thead>
<tr>
<th>IMS PCB Terminology</th>
<th>MetaSuite Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMS PCB</td>
<td>Dictionary File</td>
</tr>
<tr>
<td>IMS Segment</td>
<td>Record</td>
</tr>
<tr>
<td>Field</td>
<td>Field</td>
</tr>
<tr>
<td>IMS Index</td>
<td>Index</td>
</tr>
</tbody>
</table>

This chapter explains the following actions:
- Creating the Dictionary File (page 71)
- Defining Records (page 74)
- Defining Fields (page 77)
- Defining Indices (page 87)

For more technical information, refer to the IMS DLI File Access Guide.

11.1. Creating the Dictionary File

1. In the Tree View Window, right-click the MetaStore root icon and select Add > IMS PCB. The properties panel is displayed in the Workspace.
Two tabs are available: *Technical* and *Business*.

2. **Fill out the required fields.**

   For a detailed description of the fields, refer to the sections:
   - [Technical Tab (IMS PCB)](page 72)
   - [Business Tab (IMS PCB Data Objects)](page 90)

3. **Apply or discard your changes.**

4. **Save the changes to the MetaStore Repository.**

   In the Tree View Window, right-click the new Dictionary File and select *Save to MetaStore*.

---

**Note:** If you do not save the Dictionary File to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

---

**Technical Tab (IMS PCB)**

The following fields are available on the Technical tab:

- [Name](page 72)
- [Code-control](page 72)
- [Database](page 73)
- [Version](page 73)
- [CCSID](page 73)

### Name

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Note:** If the PCB name is not specified in the PSBGEN, or if it is spelled the same as a PCB name in another PSBGEN, you may not use it as file name.

### Code-control

Optional field.
Code-control Tables contain information about how to process a certain type of data container. Each table element of the code-control table points to a code table in the COBOL Generator Dictionary.

Each data container type has a pre-defined code-control table. If required, this default code-control table can be overwritten. For example: in order to invoke an external I/O module to read or write the file or to expand or compress data.

**Note:** When the option is used, the selected table must exist in the COBOL Generator Dictionary before programs using the file can be generated.

For more information about code-control tables, refer to the section *Code-control Tables* in the *Generator Manager Guide*.

**Database**

Enter the name of the corresponding database, if applicable.

**Note:** This field is mandatory for IMS Pcb. Enter the name of the PCB as found in the PSBGEN. Make sure that the DBDGEN job, which you are using, defines the same DBD as specified in the DBDname parameter of the PCB statement in the PSBGEN. The name of the PSB in the PSBGEN job can be found at the end of the job by the indication: `PSBNAME=PSB-name`.

**Version**

Default value = 1

You can change the version number, if required.

When reimporting or recollecting an existing data container, the version will be modified depending on the settings specified in the INI Manager. For more information, refer to the chapter MetaStore Manager Settings in the *MetaSuite INI Manager Guide*.

**CCSID**

Enter the Coded Character Set Identifier.

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.

This CCSID is an enriched property and will not be collected.

**Business Tab (IMS PCB)**

The fields on the Business Tab are identical for all IMS PCB data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section *Business Tab (IMS PCB Data Objects)* (page 90) for a description of the fields.
11.2. Defining Records

1. In the Tree View Window, right-click the appropriate IMS PCB Dictionary File and select Add IMS Segment.
   The properties panel is displayed in the Workspace.

![Image of properties panel]

Two tabs are available: Technical and Business.

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (IMS Segments) (page 74)
   - Business Tab (IMS Segments) (page 77)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Dictionary File the new Record belongs to and select Save to MetaStore.

   **Note:** If you do not add the Record to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (IMS Segments)**

The following fields are available on the Technical tab:

- Name (page 75)
- Size (page 75)
- Database (page 75)
- IMS Record Key (page 75)
- CCSID (page 76)
- Record Keys (page 76)
**Name**

Mandatory field.
The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Note:** You can use the name of the segment as specified in the DBD and PSBGENs.

**Size**

Mandatory field.
Enter the value of the BYTES parameter of the SEGM statement in the DBDGEN job.
The number specified for the record size need not be exact, but must be at least as large as the actual record size. Specifying a record size that is too small for the actual record will cause unpredictable processing results.

**Auto Calculate**

If this option is activated, the size will be calculated automatically.

**Database**

**Note:** This field is specific for each file (sub)type. Refer to the appropriate File Access Guide for more detailed information.

This field is mandatory for IMS Segments. Enter the segment name of the record, as specified in the DBDGEN and PSBGEN statements.
For RDBMS, the Creator Name should be entered in this field.

**IMS Record Key**

Select the name of the storage key of the IMS database segment, by which you can obtain the record.

1. Click the *Browse* button to display the IMS Record Key Properties panel.
2. Fill out the name of the new IMS Record Key.
3. Select the required IMS Field.
   Click the *Browse* button to display the list of all available fields and subfields.
4. Apply your changes and close the IMS Record Key Properties panel. The new Record Key is displayed in the IMS Record Key field.

**CCSID**

Enter the Coded Character Set Identifier. CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.

This CCSID is an enriched property and will not be collected.

**Record Keys**

Optional field.

This option is used to select the field, by which you can randomly obtain the segment (or record). This option must be specified for any segment whose DBDGEN definition includes a field with the "SEQ" attribute. You can determine if this is the case by examining the DBDGEN statements for the record being defined. If one of the FIELD statements, following the SEGM statement for the record in question, appears in the format below, this is the storage key for the record:

```
FIELD NAME=(IMS-field,SEQ,U),START=9999
```

1. Double-click the New icon available in the Record Keys panel.

   **Note:** You can only define Record Keys after having defined Records and Fields for the Dictionary File.

   The following screen is displayed.

2. Fill out the name of the new Key.

3. Select the required IMS Field.

   Click the Browse button to display the list of all available IMS Fields.

   Two extra options are available at the top right of the pop-up window for selecting the required item:

   - **Show all**
     
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
• Indentation
  
  When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

4. Apply your changes and close the Record Key Properties panel.
   The new Record Key is added to the Record Keys panel.

5. Repeat this action for all Record Keys you want to define.

**Business Tab (IMS Segments)**

The fields on the Business Tab are identical for all IMS PCB data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (IMS PCB Data Objects) (page 90) for a description of the fields.

### 11.3. Defining Fields

Many fields available in an IMS Segment will not be defined in the DBDGEN job. Typically, a few large group-fields will be defined with FIELD statements in the DBDGEN job, and the fine structure (the individual fields) will be defined only in the application programs. Consequently, you will need to consult with your systems staff to locate descriptions of most of the fields available in your database - the DBDGEN and PSBGEN jobs will provide only descriptions of the key fields and the large group-fields.

Various kinds of IMS "field-level sensitivity" may impact the appearance of the records returned by the DBMS. If the PSBGEN job contains SENSEG statements which specify PROCOPT=K, then only the key fields of those records may be defined to the MetaStore Manager. Likewise, if the PSBGEN job contains SENSEG statements which are followed by SENFLD statements, then only the fields defined by the SENFLD statements may be defined to the MetaStore Manager.

1. In the Tree View Window, right-click the IMS Segment you want to define new Fields for and select Add IMS Field.
   
   The properties panel is displayed in the Workspace.
2. Fill out the required field.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (IMS Fields) (page 78)
   - Business Tab (IMS PCB Data Objects) (page 90)

   **Note:** Some of the values can also be modified by double-clicking or selecting them from a drop-down list in the Record Fields Window.

3. Save or discard your changes

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Dictionary File the new Field belongs to and select *Save to MetaStore*.

   **Note:** If you do not add the Fields to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (IMS Fields)**

The following fields are available on the Technical tab:

- Properties
  - Name (page 79)
  - Size (page 79)
  - Abs. Position (page 80)
  - Rel. Position (page 80)
  - Occurs (page 80)
  - Occurrence Depending On (page 80)

- Content
  - Type (page 81)
  - Decimals (page 82)
  - Unsigned (page 83)
  - Separated and Leading (page 83)
  - Initial (page 83)
  - Code (page 83)
  - Edit Mask (page 84)
  - Date Format (page 86)
  - Low Limit (page 86)
  - High Limit (page 87)
  - Database (page 87)
  - CCSID (page 87)
**Name**

Mandatory field.
The name you enter must meet the following conditions:
- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Size**

**Note:** If the Segment field is defined in the DBDGEN job, enter the value of the BYTES parameter as the Field Size.
If the Segment field is not defined in the DBDGEN job, follow the guidelines below.

In the **Size** field, enter an integer indicating the field length as a number of bytes. If the field occurs more than once in the record, this field indicates the length of a single occurrence of the field.
This field applies for all types, except *Printed Numeric*.

If you are using a COBOL record description to define your fields, refer to the following table to determine each field’s size according to the COBOL "use" and "picture clause" definitions. This table also indicates the data type.

<table>
<thead>
<tr>
<th>COBOL Usage</th>
<th>COBOL Picture</th>
<th>MetaSuite Type</th>
<th>MetaSuite Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY</td>
<td>PIC X(c)</td>
<td>CHARACTER</td>
<td>SIZE c</td>
</tr>
<tr>
<td></td>
<td>PIC X(c)</td>
<td>HEX</td>
<td>SIZE c</td>
</tr>
<tr>
<td></td>
<td>PIC editmask</td>
<td>PRN</td>
<td>SIZE c</td>
</tr>
<tr>
<td></td>
<td>PIC S9(n)V9(d)</td>
<td>ZONED</td>
<td>SIZE n+d+s</td>
</tr>
<tr>
<td>COMPUTATIONAL</td>
<td>PIC S9(n)V9(d)</td>
<td>BINARY</td>
<td>SIZE 2 1 &lt;= n+d &lt;= 4</td>
</tr>
<tr>
<td>BINARY</td>
<td></td>
<td></td>
<td>SIZE 4 5 &lt;= n+d &lt;= 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 8 10 &lt;= n+d &lt;= 18</td>
</tr>
<tr>
<td>COMPUTATIONAL-1</td>
<td></td>
<td>FLOAT</td>
<td>SIZE 4</td>
</tr>
<tr>
<td>COMPUTATIONAL-2</td>
<td></td>
<td>FLOAT</td>
<td>SIZE 8</td>
</tr>
<tr>
<td>PACKED-DECIMAL</td>
<td>PIC S9(n)V9(d)</td>
<td>FLOAT</td>
<td>SIZE (n+d+1)/2 (rounded up)</td>
</tr>
<tr>
<td>COMPUTATIONAL-5</td>
<td>PIC S9(n)V9(d)</td>
<td>COMP-5</td>
<td>SIZE 2 1 &lt;= n+d &lt;= 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BINARY NATIVE</td>
<td>SIZE 4 5 &lt;= n+d &lt;= 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 8 10 &lt;= n+d &lt;= 18</td>
</tr>
<tr>
<td>COBOL usage NATIONAL</td>
<td>picture N(n)</td>
<td>NATIONAL</td>
<td>2*n</td>
</tr>
<tr>
<td>COBOL usage NATIONAL</td>
<td>picture 9(n)V9(d)</td>
<td>PRN-NATIONAL</td>
<td>2*(n+d)</td>
</tr>
</tbody>
</table>
Where:  
\[ c = \text{number of characters} \]
\[ n = \text{number of integer digits} \]
\[ d = \text{number of decimal digits} \]
\[ s = 1 \text{ for sign indicator (if present) or 0 for no sign indicator} \]

**Note:** The Record Layout window shows the default value for the Field within the selected Record. The default value matches the size of the Field, as defined in the Record Fields Window.

**Auto Calculate**
If this option is activated, the size will be calculated automatically.

**Abs. Position**

**Note:** If the Segment field is defined in the DBDGEN job, enter the value of the START parameter in the Size field.
If the Segment field is not defined in the DBDGEN job, follow the guidelines below.

In this field, you may change the Field's starting character position that was calculated automatically by the system. The value entered must be an integer.

**Note:** If the Record’s length is variable, you may not include the four-character record descriptor word when determining the start position of the Field.

**Rel. Position**

This is the 1-based position of this field within its domed structure (i.e., Record or Group).

- For a field that is not a subfield, the absolute and relative positions are equal.
- For a field x that is a subfield of group g, the relative position will be calculated as 1 + (absolute position of x) - (absolute position of g).

For example: if group g starts on position 5, and the absolute position of x is also 5, then the relative position of x is 1 + 5 - 5 = 1. In human terms: "x will start on the 1st position of g".

**Occurs**

Optional field.
Enter an integer indicating the maximum number of times this field can occur in the record. If you do not define a specific value, the field is assumed to occur once.

**Occurrence Depending On**

Optional field. Only applies for repeating fields (n).
Select a numeric field that is defined prior to this field within the record.

Results:
- The value you defined with the *Occurs* option represents the *maximum* number of times this field can occur.
- The value available in the selected field represents the *actual* number of times the field occurs.

1. Click the *Browse* button next to the *Occurrence Depending On* text field.
   The list of available numeric fields is displayed.

2. Select the required Field.
   The name of the selected Field is displayed in the *Occurrence Depending On* text field.

   **Note:** If you click the *Browse* button again now, the Properties screen for the selected field will be displayed. If you want to select another field, you first have to delete the current field name from the text field and then click the *Browse* button.

**Type**

This field is mandatory and contains the default value *Character*.

There are two general classes of data types: *non-numeric* and *numeric*.

**Non-numeric Data Types**

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td>This datatype indicates a field containing strings using the characters A-Z and a-z.</td>
</tr>
<tr>
<td>Character</td>
<td>This datatype indicates that the field may contain any possible character within the character set being used (including unprintable characters). It is not possible to perform numeric operations on a Character type field, even when the field contains only digits. Matching RDBMS Data Types: CHAR, BYTE, VARCHAR, VARCHAR2, LONG, RAW, LONGVARCHAR</td>
</tr>
<tr>
<td>Graphic</td>
<td>DB2 terminology for Character datatype.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>This datatype indicates that any hexadecimal characters are allowed within the field.</td>
</tr>
<tr>
<td>Long Varchar</td>
<td>This datatype indicates a field containing a variable character string of maximum 2 GB.</td>
</tr>
<tr>
<td>National</td>
<td>The National datatype is a subset of Unicode. The character set on which it is based is UTF-16, but restricted to two bytes per character.</td>
</tr>
<tr>
<td>Varchar</td>
<td>This datatype indicates a character field allowing the storage of values you need plus one byte for the length.</td>
</tr>
<tr>
<td>Vargraphic</td>
<td>DB2 terminology for Varchar datatype.</td>
</tr>
</tbody>
</table>
## Numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>This datatype indicates that the field contains a binary numeric value composed of 0's and 1's.</td>
</tr>
<tr>
<td>Binary Native</td>
<td>This datatype indicates that the field contains binary native numbers. This means that its internal representation depends on the Operating System and/or processor.</td>
</tr>
<tr>
<td>Bit</td>
<td>This datatype indicates a field occupying a single bit storage. A BIT type field may only contain the binary values 0 or 1. It is possible to perform numeric operations on a BIT type field.</td>
</tr>
<tr>
<td>Byte</td>
<td>This datatype indicates a field occupying a single byte storage. A byte is 8 bits. It is possible to perform numeric operations on a BYTE type field.</td>
</tr>
<tr>
<td>Decimal</td>
<td>This datatype indicates a field containing a packed decimal value. Packed decimal is the most commonly used internal numeric data type. Two decimal digits are contained in each byte of a packed decimal number. However, if the number is signed, the last half byte contains a positive or negative sign indicator.</td>
</tr>
<tr>
<td>Float</td>
<td>This datatype indicates that the field contains floating-point numbers, encoded in an encoded exponential form.</td>
</tr>
<tr>
<td>Numeric</td>
<td>This datatype indicates that any the field contains decimal numbers in a printable character format, this means that a single digit is stored per byte. Leading blanks are not permitted in this field type, nor may it contain editing characters, such as commas or decimal points. Use Printed Numeric in this case. Matching RDBMS Data Types: TINYINT, SMALLINT, INTEGER, BIGINT, DECIMAL, NUMBER, FLOAT</td>
</tr>
<tr>
<td>Printed Numeric</td>
<td>This datatype indicates that the character-based field contains a numeric value. The format in which the printed numeric value is displayed must be specified with the EDIT option.</td>
</tr>
<tr>
<td>Printed Numeric National</td>
<td>This datatype indicates that this is a National field that contains a numeric value. The format in which the PRN-NATIONAL value is displayed must be specified with the EDIT option.</td>
</tr>
</tbody>
</table>

### Decimals

If the field type allows the definition of decimals, this field contains the number of decimals. Decimals can be entered for the following field types:

- Binary
- Binary Native
- Decimal
- Numeric
- Printed Numeric
- Printed Numeric National

If the field type does not allow the definition of decimals, this field contains the default value 0.
Unsigned

Select the Unsigned check box to indicate that a numeric value is not signed, i.e. that it does not contain a sign indicator (+ or -).
This field applies for the following field types:
- Binary
- Binary Native
- Numeric

Note: For Decimal fields, the distinction between unsigned and positive signed INPUT fields is not relevant to the system.

Separated and Leading

These two check boxes allow to define how over-punched and separate sign indicators must be treated. They only apply for the Numeric field type.
There are several possibilities.
If the sign indicator is over-punched in the number, select the Unsigned check box above.
- Select Leading to put the sign on the initial digit.
- Select Separate to put the sign on the last byte (no digit).
- Select Leading and Separate to put the sign in the initial byte (no digit).
If the Numeric field contains a separate plus or minus sign attached to the number, select the Separate check box.

Initial

Optional field.
Enter the initial value for the field.
This value is only taken into account for Target Files. If an initial value was defined for a Source file, this parameter will not affect any operation, except for syntax checking.
For Numeric fields, this is a numeric constant, where the decimal point should be given by a . (point).
For Character fields, this may be an alphanumeric constant (not enclosed by quotes) or the system fields SYS-LOW-VALUE or SYS-HIGH-VALUE, which correspond to system fields LOW-VALUE and HIGH-VALUE within COBOL.

Code

Default value = No Code
The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>This field is treated as a code, not as a number.</td>
</tr>
</tbody>
</table>
**Edit Mask**

This field is mandatory for Printed Numeric fields, but optional for other field types. It indicates how the alphanumeric values must be formatted.

**Note:** When the Edit Mask is set for a printed numeric field, the MetaStore Manager will reset the size of the field to the size that corresponds to the chosen Edit Mask.

Enter the characters defining a Mask in this text field. This Mask will override the default mask for the field. There is a default mask for each field type. Both the default masks and the manually created masks are composed of Replacement and Insertion characters.

The following table lists the Replacement characters and their meaning. Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field.

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>N</td>
<td>National (2-byte Unicode character set)</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
</tbody>
</table>

The following table lists the Insertion characters and their meaning. Insertion characters indicate characters to be printed in addition to those contained in the stored field.

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
</tbody>
</table>
As mentioned above, there is a default Mask for each field type:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Default Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields</td>
<td>The default mask contains a minus sign as the rightmost character. All negative values are printed with a trailing minus sign.</td>
</tr>
<tr>
<td>Numeric fields with decimals</td>
<td>The default mask contains a decimal point and as many digit replacement characters (9s) to the right of the decimal point as are specified by the Decimal option.</td>
</tr>
<tr>
<td>Numeric fields</td>
<td>The default mask contains as many digits as its size, without zero suppression.</td>
</tr>
<tr>
<td>Date fields</td>
<td>The default mask is its selected date format.</td>
</tr>
<tr>
<td>Alphanumeric fields</td>
<td>The default mask contains as many alphanumeric character replacement characters (X) as are required to print the field.</td>
</tr>
</tbody>
</table>

Examples of default masks:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Size</th>
<th>Default mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields without decimal positions</td>
<td>6</td>
<td>999999-</td>
</tr>
<tr>
<td>Signed numeric fields with two decimals</td>
<td>6</td>
<td>9999.99-</td>
</tr>
<tr>
<td>Character Field</td>
<td>6</td>
<td>X(6)</td>
</tr>
<tr>
<td>Unicode Field</td>
<td>6</td>
<td>N(3)</td>
</tr>
</tbody>
</table>

You may also define customized masks. Examples:
Date Format

If the field must contain a date, select the required date format from the drop-down list. The system automatically validates date fields whenever they are referenced in a MetaMap model, and automatically converts date fields whenever they are compared to another date or used in a calculation. In all the available formats, YY or YYYY stands for the year and MM stands for the month. DD stands for the day within a month and DDD stands for the day within the year.

When the format contains a ‘?’ , this indicates the date delimiter that is used. Data formats with a ‘?’ are only supported for Character and Varchar field types. When the data format does not contain a ‘?’ , the different parts in the date are not delimited by a special character.

The Date Format list is accessible for the following field types:

- Binary
- Binary Native
- Character
- Decimal
- Numeric
- National
- Varchar

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>6</td>
<td>XXBXXXX</td>
<td>AB138B</td>
<td>AB 138B</td>
</tr>
<tr>
<td>Numeric with 2 decimal positions</td>
<td>2</td>
<td>.99</td>
<td>.35 0 -.12</td>
<td>.35 .00 .12</td>
</tr>
<tr>
<td>Numeric with 3 minus signs</td>
<td>4</td>
<td>---9</td>
<td>0 -1 210</td>
<td>0 -1 210</td>
</tr>
<tr>
<td>Numeric with 3 plus signs</td>
<td>4</td>
<td>+++9</td>
<td>0 -1 210</td>
<td>+0 -1 210</td>
</tr>
</tbody>
</table>

Note: When a date format is chosen, MetaStore Manager will reset the size of the field to the size that corresponds to the chosen date format.

Low Limit

Optional field.

When a Low Limit is specified, you must define a value in the High Limit field as well.

You may use this field to define a minimum value for this field. If a lower value is encountered, the system will consider the data invalid.
High Limit

Optional field.
When a High Limit is specified, you must define a value in the Low Limit field as well. You may use this field to define a maximum value for this field. If a higher value is encountered, the system will consider the data invalid.

Database

This field contains the IMS name which is needed for key and index fields. If this field is not a key field nor an index field, this database property can be omitted.
For more technical information, refer to the IMS DLI File Access Guide or contact your IMS DLI database administrator.

CCSID

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.

Business Tab (IMS Fields)

The fields on the Business Tab are identical for all IMS PCB data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (IMS PCB Data Objects) (page 90) for a description of the fields.

11.4. Defining Indices

Indices need to be defined if the database is an indexed database (HISAM, HIDAM, etc.) or if the database has secondary indexes defined.
1. In the Tree View Window, right-click the new IMS PCB Dictionary File you want to define an Index for and select Add IMS Index from the context menu. The properties panel is displayed in the Workspace.

2. Fill out the required fields on the Technical Tab. For a detailed description of the fields, refer to the sections:
   - Technical Tab (IMS Indices) (page 88)
   - Business Tab (IMS PCB Data Objects) (page 90)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository. In the Tree View Window, right-click the Dictionary File the new Index belongs to and select Save to MetaStore.

   **Note:** If you do not add the Index to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (IMS Indices)**

The following fields are available on the Technical tab:
- Name (page 88)
- Database (page 89)
- Index Keys (page 89)

**Name**

Mandatory field. The name you enter must meet the following conditions:
- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
It may contain the characters a-z, A-Z and 0-9.
• It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
• It may not be ended with a – (hyphen).

Database

Each IMS index is an IMS database, called Index Database. The Segments (or Records) are not accessed by a MetaSuite Program, but by the DBMS, which controls the sequence to the database. As this is a transparent operation, there is no need to specify the Database Name.

Index Keys

Mandatory field. Records in an IMS database are accessed by using Pointers. This field contains the Pointer that you want to use in order to access the Records. The IMS field names will be specified as the internal database names when we define these MetaSuite fields later.

1. Double-click the New icon available in the Index Keys panel. A screen similar to this one is displayed:

2. First select the Item (IMS Segment or IMS Field) using the drop-down list. All available fields for the selected IMS Segment or IMS Field will be displayed underneath. Two extra options are available at the top right of the pop-up window for selecting the required item:
   • Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   • Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.
3. Select the required Field and click OK.
   The IMS Index Key properties panel is displayed.

   ![IMS Index Key properties panel]

4. Fill out the required fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>The name of the new Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Displays the name of the Field you selected.</td>
</tr>
<tr>
<td></td>
<td>Click the button to display its properties.</td>
</tr>
<tr>
<td></td>
<td>Click the button to remove the Field and select another one.</td>
</tr>
</tbody>
</table>

5. Apply your changes and close the Index Key Properties panel.
   The Index Key name is added in the Index Keys panel.

6. Repeat this action for all Index Keys you want to define.

7. If you need to change the order of the Index Keys:
   - Right-click the key to be moved.
   - Select Move Forward or Move Backward from the pop-up menu, until the Key reaches its required position.

   **Note:** You can also create and delete Index Keys using this pop-up menu.

### Business Tab (IMS Indices)

The fields on the Business Tab are identical for all IMS PCB data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (IMS PCB Data Objects) (page 90) for a description of the fields.

### 11.5. Business Tab (IMS PCB Data Objects)

The following fields are available on the Business tab:

- **Business Rule** (page 91)
- **Note** (page 91)

   **Note:** If you want to enter text in RTF (Rich Text Format), right-click and select RTF from the context menu (or use the shortcut CTRL + R).
Business Rule
Optional field.
Enter a free-form description for the data object (Dictionary File, Record, Field, Relationship or Index). For example, the business rule or the requirements.

Note
Optional field.
Enter a free-form note for the data object.
CHAPTER 12

SQL Table Group

This chapter explains the following actions:

- Creating the Dictionary File (page 92)
- Defining Records (page 97)
- Defining Fields (page 99)

For more technical information, refer to the RDBMS File Access Guide.

12.1. Creating the Dictionary File

1. In the Tree View Window, right-click the MetaStore root icon and select Add > SQL Table Group. The properties panel is displayed in the Workspace.

![Technical and Business tabs](image)

Two tabs are available: Technical and Business.

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (SQL Table Groups) (page 93)
   - Business Tab (SQL Table Group Data Objects) (page 108)
3. Apply or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the new SQL Table Group and select *Save to MetaStore*.

**Note:** If you do not save the SQL Table Group to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (SQL Table Groups)**

The following fields are available on the Technical tab:

- **Name** (page 93)
- **Subtype** (page 93)
- **Code-control** (page 94)
- **Database** (page 94)
- **Version** (page 94)
- **CCSID** (page 94)
- **Fetch First Row Only** (page 95)
- **Uncommitted Read** (page 95)
- **File Keys** (page 95)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Subtype**

Default value = *Generator_Default*

Select the required Subtype from the drop-down list. These options match the RDBMS you are working with.

The following options are available:

- **DB2 for OS/400**
- **DB2 for z/OS**
- **DB2 LUW**
- **DB2/2**
Code-control

Optional field.
Code-control Tables contain information about how to process a certain type of data container. Each table element of the code-control table points to a code table in the COBOL Generator Dictionary. Each data container type has a pre-defined code-control table. If required, this default code-control table can be overwritten. For example: in order to invoke an external I/O module to read or write the file or to expand or compress data.

**Note:** When the option is used, the selected table must exist in the COBOL Generator Dictionary before programs using the file can be generated.

For more information about code-control tables, refer to the section Code-control Tables in the Generator Manager Guide.

Database

Enter the name of the corresponding database, if applicable.

Version

Default value = 1
You can change the version number, if required.
When reimporting or recollecting an existing data container, the version will be modified depending on the settings specified in the INI Manager. For more information, refer to the chapter MetaStore Manager Settings in the MetaSuite INI Manager Guide.

CCSID

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.

This CCSID is an enriched property and will not be collected.

**Fetch First Row Only**

Select this check box, if you want to read only the first row, in case different rows contain the same key.

**Uncommitted Read**

Select this check box, if you want to perform a so-called dirty read. This means reading data without checking if the data has been locked or not.

**File Keys**

Optional field.

This field is available for Standard Files with subtype *Record Sequential*, *Index* or *VSAM*, and for SQL Table Groups.

**Note:** You can only define File Keys after having defined Data Structures (Standard Records or SQL Tables) and Data Entities (Standard Fields or SQL Columns).

1. Double-click the *New* icon available in the *File Keys* panel.

   A screen similar to this one is displayed:

   ![Screen with File Keys](image)

2. First select the *Item* (the SQL Table) using the drop-down list.

   All available Columns for the selected Table will be displayed underneath.
3. Select the required SQL Column and click OK. The *SQL Table Group Key* properties panel is displayed.

![SQL Table Group Key properties panel](image)

4. Fill out the required fields.

   **Name**
   - By default, this field contains the name of the Data Entity you selected. If required you can modify the name.

   **Column**
   - Displays the name of the Column you selected.
   - Click the button to display its properties.
   - Click the button to remove the Column and select another one.

5. Apply your changes and close the *SQL Table Group Key Properties* panel. The new File Key is added to the *File Keys* panel.

6. Repeat this action for all File Keys you want to define.

7. If you need to change the order of the File Keys:
   - Right-click the key to be moved.
   - Select *Move Forward* or *Move Backward* from the pop-up menu, until the Key reaches its required position.

   **Note:** You can also create and delete File Keys using this pop-up menu.

**Business Tab (SQL Table Groups)**

The fields on the Business Tab are identical for all SQL Table Group data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section *Business Tab (SQL Table Group Data Objects)* (page 108) for a description of the fields.
12.2. Defining Records

1. In the Tree View Window, right-click the SQL Table Group you want to define Records for and select *Add SQL Table.*

   The properties panel is displayed in the Workspace.

   ![Properties Panel]

   Two tabs are available: *Technical* and *Business.*

2. Fill out the required fields.

   For a detailed description of the fields, refer to the sections:
   - Technical Tab (SQL Tables) (page 97)
   - Business Tab (SQL Table Group Data Objects) (page 108)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.

   In the Tree View Window, right-click the SQL Table Group the new Table belongs to and select *Save to MetaStore.*

   **Note:** If you do not add the Table to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (SQL Tables)**

The following fields are available on the Technical tab:

- **Name** (page 97)
- **Creator** (page 98)
- **Location** (page 98)
- **Type** (page 98)
- **CCSID** (page 98)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Creator**

Enter the name of the Table Creator, as defined in the relational catalogue. If omitted, the value will be considered being *Null*.

**Location**

This field has become obsolete.

**Type**

This field has become obsolete.

**CCSID**

Enter the Coded Character Set Identifier. CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page. CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL. This CCSID is an enriched property and will not be collected.

**Business Tab (SQL Tables)**

The fields on the Business Tab are identical for all SQL Table Group data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section **Business Tab (SQL Table Group Data Objects)** (page 108) for a description of the fields.
12.3. Defining Fields

1. In the Tree View Window, double-click the Record you want to define Fields for and select Add SQL Column.
   
   The properties panel is displayed in the Workspace.

   ![Properties Panel](image)

   Two tabs are available: Technical and Business.

2. Fill out the required field.
   
   For a detailed description of the fields, refer to the sections:
   
   - Technical Tab (SQL Columns) (page 100)
   - Business Tab (SQL Table Group Data Objects) (page 108)

   **Note:** Some of the values can also be modified by double-clicking or selecting them from a drop-down list in the Record Fields Window.

3. Save or discard your changes

4. Save the changes to the MetaStore Repository.
   
   In the Tree View Window, right-click the SQL Table Group the new Column belongs to and select Save to MetaStore.

   **Note:** If you do not add the Column to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.
Technical Tab (SQL Columns)

The following fields are available on the Technical tab:

- **Properties**
  - Name (page 100)
  - Size (page 102)

- **Content**
  - Type (page 100)
  - Decimals (page 103)
  - Unsigned (page 103)
  - Separated and Leading (page 103)
  - Nulls Allowed (page 104)
  - With Default (page 104)
  - Date Format (page 104)
  - Edit Mask (page 105)
  - Code (page 107)
  - Low Limit (page 107)
  - High Limit (page 107)
  - CCSID (page 107)

**Name**

Mandatory field.
The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a–z, A–Z and 0–9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Type**

This field is mandatory and contains the default value Character.
There are two general classes of data types: non-numeric and numeric.

**Non-numeric Data Types**

<table>
<thead>
<tr>
<th>Alphabetic</th>
<th>This datatype indicates a field containing strings using the characters A-Z and a-z.</th>
</tr>
</thead>
</table>
### Numeric Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>This datatype indicates that the field may contain any possible character within the character set being used (including <em>unprintable</em> characters). It is not possible to perform numeric operations on a Character type field, even when the field contains only digits. Matching RDBMS Data Types: CHAR, BYTE, VARCHAR, VARCHAR2, LONG, RAW, LONGVARCHAR</td>
</tr>
<tr>
<td>Graphic</td>
<td>DB2 terminology for Character datatype.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>This datatype indicates that any hexadecimal characters are allowed within the field.</td>
</tr>
<tr>
<td>Long Varchar</td>
<td>This datatype indicates a field containing a variable character string of maximum 2 GB.</td>
</tr>
<tr>
<td>National</td>
<td>The National datatype is a subset of Unicode. The character set on which it is based is UTF-16, but restricted to two bytes per character.</td>
</tr>
<tr>
<td>Varchar</td>
<td>This datatype indicates a character field allowing the storage of values you need plus one byte for the length.</td>
</tr>
<tr>
<td>Vargraphic</td>
<td>DB2 terminology for Varchar datatype.</td>
</tr>
<tr>
<td>Binary</td>
<td>This datatype indicates that the field contains a binary numeric value composed of 0's and 1's.</td>
</tr>
<tr>
<td>Binary Native</td>
<td>This datatype indicates that the field contains binary native numbers. This means that its internal representation depends on the Operating System and/or processor.</td>
</tr>
<tr>
<td>Bit</td>
<td>This datatype indicates a field occupying a single bit storage. A BIT type field may only contain the binary values 0 or 1. It is possible to perform numeric operations on a BIT type field.</td>
</tr>
<tr>
<td>Byte</td>
<td>This datatype indicates a field occupying a single byte storage. A byte is 8 bits. It is possible to perform numeric operations on a BYTE type field.</td>
</tr>
<tr>
<td>Decimal</td>
<td>This datatype indicates a field containing a packed decimal value. Packed decimal is the most commonly used internal numeric data type. Two decimal digits are contained in each byte of a packed decimal number. However, if the number is signed, the last half byte contains a positive or negative sign indicator.</td>
</tr>
<tr>
<td>Float</td>
<td>This datatype indicates that the field contains floating-point numbers, encoded in an encoded exponential form.</td>
</tr>
<tr>
<td>Numeric</td>
<td>This datatype indicates that any the field contains decimal numbers in a printable character format, this means that a single digit is stored per byte. Leading blanks are not permitted in this field type, nor may it contain editing characters, such as commas or decimal points. Use <em>Printed Numeric</em> in this case. Matching RDBMS Data Types: TINYINT, SMALLINT, INTEGER, BIGINT, DECIMAL, NUMBER, FLOAT</td>
</tr>
</tbody>
</table>
Printed Numeric
This datatype indicates that the character-based field contains a numeric value. The format in which the printed numeric value is displayed must be specified with the EDIT option.

Printed Numeric National
This datatype indicates that this is a National field that contains a numeric value. The format in which the PRN-NATIONAL value is displayed must be specified with the EDIT option.

**Size**

In the **Size** field, enter an integer indicating the field length as a number of bytes. If the field occurs more than once in the record, this field indicates the length of a single occurrence of the field.

This field applies for all types, except Printed Numeric.

If you are using a COBOL record description to define your fields, refer to the following table to determine each field’s size according to the COBOL “use” and “picture clause” definitions. This table also indicates the data type.

<table>
<thead>
<tr>
<th>COBOL Usage</th>
<th>COBOL Picture</th>
<th>MetaSuite Type</th>
<th>MetaSuite Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY</td>
<td>PIC X(c)</td>
<td>CHARACTER</td>
<td>SIZE c</td>
</tr>
<tr>
<td></td>
<td>PIC X(c)</td>
<td>HEX</td>
<td>SIZE c</td>
</tr>
<tr>
<td></td>
<td>PIC editmask</td>
<td>PRN</td>
<td>SIZE c</td>
</tr>
<tr>
<td></td>
<td>PIC S9(n)V9(d)</td>
<td>ZONED</td>
<td>SIZE n+d+s</td>
</tr>
<tr>
<td>COMPUTATIONAL BINARY</td>
<td>PIC S9(n)V9(d)</td>
<td>BINARY</td>
<td>SIZE 2 1 &lt;= n+d &lt;= 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 4 5 &lt;= n+d &lt;= 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 8 10 &lt;= n+d &lt;= 18</td>
</tr>
<tr>
<td>COMPUTATIONAL-1</td>
<td>FLOAT</td>
<td></td>
<td>SIZE 4</td>
</tr>
<tr>
<td>COMPUTATIONAL-2</td>
<td>FLOAT</td>
<td></td>
<td>SIZE 8</td>
</tr>
<tr>
<td>COMPUTATIONAL-3</td>
<td>PIC S9(n)V9(d)</td>
<td>FLOAT</td>
<td>SIZE (n+d+1)/2 (rounded up)</td>
</tr>
<tr>
<td>PACKED-DECIMAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPUTATIONAL-5</td>
<td>PIC S9(n)V9(d)</td>
<td>BINARY NATIVE</td>
<td>SIZE 2 1 &lt;= n+d &lt;= 4</td>
</tr>
<tr>
<td></td>
<td>COMP-5</td>
<td></td>
<td>SIZE 4 5 &lt;= n+d &lt;= 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 8 10 &lt;= n+d &lt;= 18</td>
</tr>
<tr>
<td>COBOL usage NATIONAL</td>
<td>picture N(n)</td>
<td>NATIONAL</td>
<td>2*n</td>
</tr>
<tr>
<td>COBOL usage NATIONAL</td>
<td>picture 9(n)V9(d)</td>
<td>PRN-NATIONAL</td>
<td>2*(n+d)</td>
</tr>
</tbody>
</table>
Where:  
\[ \begin{align*} 
    c &= \text{number of characters} \\
    n &= \text{number of integer digits} \\
    d &= \text{number of decimal digits} \\
    s &= 1 \text{ for sign indicator (if present) or 0 for no sign indicator} 
\end{align*} \]

Note: The Record Layout window shows the default value for the Field within the selected Record. The default value matches the size of the Field, as defined in the Record Fields Window.

Auto Calculate
If this option is activated, the size will be calculated automatically.

Decimals
If the field type allows the definition of decimals, this field contains the number of decimals.

Decimals can be entered for the following field types:
- Binary
- Binary Native
- Decimal
- Numeric
- Printed Numeric
- Printed Numeric National

If the field type does not allow the definition of decimals, this field contains the default value 0.

Unsigned
Select the Unsigned check box to indicate that a numeric value is not signed, i.e. that it does not contain a sign indicator (+ or -).

This field applies for the following field types:
- Binary
- Binary Native
- Numeric

Note: For Decimal fields, the distinction between unsigned and positive signed INPUT fields is not relevant to the system.

Separated and Leading
These two check boxes allow to define how over-punched and separate sign indicators must be treated. They only apply for the Numeric field type.

There are several possibilities.
If the sign indicator is over-punched in the number, select the *Unsigned* check box above.

- Select *Leading* to put the sign on the initial digit.
- Select *Separate* to put the sign on the last byte (no digit).
- Select *Leading* and *Separate* to put the sign in the initial byte (no digit).

If the Numeric field contains a separate plus or minus sign attached to the number, select the *Separate* check box.

### Nulls Allowed

If you select the *Nulls allowed* option, no DBNAME statement will be added to the field description when exporting to MDL.

If you select the *Nulls allowed* option, the field will be considered as being "INNULL".

If you unflag the *Nulls allowed* option, the field will be considered as being "NOTNULL".

### With Default

By default this option is selected, which means that default NULLS are not allowed.

When exporting to MDL, the statement `DBNAME 'DEFAULT'` will be added to the field description.

**Note:** If you uncheck both the *Nulls allowed* and the *With default* option, the statement `DBNAME 'NOTNULL'` will be added to the field description when exporting to MDL.

### Date Format

If the field must contain a date, select the required date format from the drop-down list. The system automatically validates date fields whenever they are referenced in a MetaMap model, and automatically converts date fields whenever they are compared to another date or used in a calculation. In all the available formats, YY or YYYY stands for the year and MM stands for the month. DD stands for the day within a month and DDD stands for the day within the year.

When the format contains a '?', this indicates the date delimiter that is used. Data formats with a '?' are only supported for Character and Varchar field types. When the data format does not contain a '?', the different parts in the date are not delimited by a special character.

The Date Format list is accessible for the following field types:

- Binary
- Binary Native
- Character
- Decimal
- Numeric
- National
- Varchar

**Note:** When a date format is chosen, MetaStore Manager will reset the size of the field to the size that corresponds to the chosen date format.
**Edit Mask**

This field is mandatory for Printed Numeric fields, but optional for other field types. It indicates how the alphanumeric values must be formatted.

**Note:** When the Edit Mask is set for a printed numeric field, the MetaStore Manager will reset the size of the field to the size that corresponds to the chosen Edit Mask.

Enter the characters defining a Mask in this text field. This Mask will override the default mask for the field. There is a default mask for each field type. Both the default masks and the manually created masks are composed of *Replacement* and *Insertion* characters.

The following table lists the Replacement characters and their meaning. Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field.

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>N</td>
<td>National (2-byte Unicode character set)</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
</tbody>
</table>

The following table lists the Insertion characters and their meaning. Insertion characters indicate characters to be printed in addition to those contained in the stored field.

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
<tr>
<td>,</td>
<td>Comma (separator for the sake of large number readability or decimal separator depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>.</td>
<td>Decimal point (separator for the sake of large number readability or decimal separator, depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>B</td>
<td>Blank</td>
</tr>
<tr>
<td>-</td>
<td>Floating or trailing minus for negative values. Plus is not accepted as valid character. Positive numbers will have a blank as sign character.</td>
</tr>
<tr>
<td>+</td>
<td>Floating or trailing plus or minus sign</td>
</tr>
<tr>
<td>CR</td>
<td>Trailing credit symbol for negative values only</td>
</tr>
</tbody>
</table>
As mentioned above, there is a default Mask for each field type:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Default Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields</td>
<td>The default mask contains a minus sign as the rightmost character. All negative values are printed with a trailing minus sign.</td>
</tr>
<tr>
<td>Numeric fields with decimals</td>
<td>The default mask contains a decimal point and as many digit replacement characters (9s) to the right of the decimal point as are specified by the Decimal option.</td>
</tr>
<tr>
<td>Numeric fields</td>
<td>The default mask contains as many digits as its size, without zero suppression.</td>
</tr>
<tr>
<td>Date fields</td>
<td>The default mask is its selected date format.</td>
</tr>
<tr>
<td>Alphanumeric fields</td>
<td>The default mask contains as many alphanumeric character replacement characters (X) as are required to print the field.</td>
</tr>
</tbody>
</table>

Examples of default masks:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Size</th>
<th>Default mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields without decimal positions</td>
<td>6</td>
<td>999999-</td>
</tr>
<tr>
<td>Signed numeric fields with two decimals</td>
<td>6</td>
<td>9999.99-</td>
</tr>
<tr>
<td>Character Field</td>
<td>6</td>
<td>X(6)</td>
</tr>
<tr>
<td>Unicode Field</td>
<td>6</td>
<td>N(3)</td>
</tr>
</tbody>
</table>

You may also define customized masks. Examples:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>6</td>
<td>XXBXXX</td>
<td>AB138B</td>
<td>AB 138B</td>
</tr>
<tr>
<td>Numeric with 2 decimal positions</td>
<td>2</td>
<td>.99</td>
<td>.35</td>
<td>.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>.0</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-.12</td>
<td>-.12</td>
</tr>
<tr>
<td>Numeric with 3 minus signs</td>
<td>4</td>
<td>---9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>
Code

Default value = No Code
The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>This field is treated as a code, not as a number.</td>
</tr>
<tr>
<td>No Code</td>
<td>Resets the code operator to zero. Select this option when there is no additional information to be added for the field.</td>
</tr>
<tr>
<td>Time</td>
<td>Select this option to define a field that contains TIME information.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Select this option to define a field that contains TIMESTAMP information.</td>
</tr>
</tbody>
</table>

Low Limit

Optional field.
When a Low Limit is specified, you must define a value in the High Limit field as well.
You may use this field to define a minimum value for this field. If a lower value is encountered, the system will consider the data invalid.

High Limit

Optional field.
When a High Limit is specified, you must define a value in the Low Limit field as well. You may use this field to define a maximum value for this field. If a higher value is encountered, the system will consider the data invalid.

CCSID

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.
12.4. Business Tab (SQL Table Group Data Objects)

The following fields are available on the Business tab:

- **Business Rule** (page 108)
- **Note** (page 108)

**Note:** If you want to enter text in RTF (Rich Text Format), right-click and select RTF from the context menu (or use the shortcut CTRL + R).

### Business Rule

Optional field.

Enter a free-form description for the data object (Dictionary File, Record, Field, Relationship or Index). For example, the business rule or the requirements.

### Note

Optional field.

Enter a free-form note for the data object.
A file is considered to be a "Standard File", if the data is stored in physical files that are not controlled by a DBMS (database management system).

This chapter explains the following actions:

- Creating the Dictionary File (page 109)
- Defining Records (page 117)
- Defining Fields (page 122)

### 13.1. Creating the Dictionary File

1. In the Tree View Window, right-click the MetaStore root icon and select Add > Standard File. The properties panel is displayed in the Workspace.

Two tabs are available: Technical and Business.

2. Fill out the required fields.

   For a detailed description of the fields, refer to the sections:

   - Technical Tab (Standard Files) (page 110)
   - Business Tab (Standard File Data Objects) (page 133)
3. Apply or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the new Standard File and select Save to MetaStore.

   **Note:** If you do not save the Standard File to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

### Technical Tab (Standard Files)

The following fields are available on the Technical tab:

- **Name** (page 110)
- **Subtype** (page 111)
- **Code-control** (page 112)
- **Database** (page 112)
- **Version** (page 112)
- **CCSID** (page 113)
- **Recording Mode** (page 113)
- **Record Format** (page 113)
- **Label** (page 114)
- **File Description Size** (page 114)
- **Block Size** (page 114)
- **Column Separator** (page 115)
- **Row Terminator** (page 115)
- **External Source** (page 115)
- **Spanned** (page 115)
- **File Keys** (page 116)

### Name

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a–z, A–Z and 0–9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.
Subtype

Select the required Subtype from the drop-down list.
Default value = Sequential

<table>
<thead>
<tr>
<th>Subtype</th>
<th>Description</th>
</tr>
</thead>
</table>
| Delimited     | Select this option, if the file is organized as sequential and if all field values in the file are separated by a predefined token. This predefined token is also known as "the delimiter". Delimited files are often saved on Windows systems with the extension ".csv", which is the abbreviation of "comma separated values". The field values within the delimited file have following restrictions:  
  • Numeric field values: any digit sequence with a "." or "," as decimal point depending on the decimal point default of the MetaSuite COBOL Generator library, with a prefixed sign "+" or "−".  
  • Character field values: may be enclosed in single quotes or double quotes or no quotes at all.                                                                                                                                                                                                                                                                                         |
| Function      | Select this option if you want to describe a structure that is not part of an input-output process, for instance the input and output fields required for a subroutine.                                                                                                                                                                                                                                                                                                                                                         |
| Index         | Select this option if the records of the file are sequenced logically (but not necessarily physically) according to the value of a specified key field. Records of this type of file may be accessed sequentially in key sequence, or randomly, for specific values of the key field. The key field for a TYPE INDEX file must be identified using the File Keys (page 116) option. In Mainframe terminology, this type of file organization is referred to as ISAM (Indexed Sequential Access Method).                                                                                                                                                                                                 |
| Line Sequential| Used on open systems (Windows, UNIX and Linux). Select this option if the records of the file are physically stored in sequential order and may only be retrieved in the order in which they are written. Each record is separated by a delimiter, which is the Carriage Return/Line Feed character.                                                                                                                                                                                                                                           |
| MetaSuite File System | This option is reserved for future use.                                                                                                                                                                                                                                                                                                                                                                                             |
| MQ-Series     | WebSphere MQ (formerly MQ-Series) is the standard for messaging across multiple platforms. Select this option if you want to store such messages in the MetaStore Repository.                                                                                                                                                                                                                                                                                                                                                     |
| Record Sequential | Used on mainframes (Z/OS, BS2000 etc.). Select this option if the records of the file are physically stored in sequential order and may only be retrieved in the order in which they are written. Each record is identified by its position within the file, and is not separated by a special delimiter. The length of each record is fixed or variable. If the length is variable, the value is put in the four-byte Record Descriptor word that is put before each record.                                                                                                                                                                                                                     |
| Relative      | Select this option if the file is a relative record file in which records are stored and retrieved according to the value of a relative record number (the number of the record relative to the beginning of the file). In IBM terminology, this type of file organization is referred to as BDAM (Basic Direct Access Method).                                                                                                                                                                                                                                         |
Code-control

Optional field.

Code-control Tables contain information about how to process a certain type of data container. Each table element of the code-control table points to a code table in the COBOL Generator Dictionary.

Each data container type has a pre-defined code-control table. If required, this default code-control table can be overwritten. For example: in order to invoke an external I/O module to read or write the file or to expand or compress data.

**Note:** When the option is used, the selected table must exist in the COBOL Generator Dictionary before programs using the file can be generated.

For more information about code-control tables, refer to the section Code-control Tables in the Generator Manager Guide.

Database

Enter the name of the corresponding database, if applicable.

Version

Default value = 1

You can change the version number, if required.
When reimporting or recollecting an existing data container, the version will be modified depending on the settings specified in the INI Manager. For more information, refer to the chapter MetaStore Manager Settings in the MetaSuite INI Manager Guide.

**CCSID**

Enter the Coded Character Set Identifier.

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.

This CCSID is an enriched property and will not be collected.

**Recording Mode**

Select the Recording Mode from the drop-down list.

Default value = **Native**

<table>
<thead>
<tr>
<th>Recording Mode</th>
<th>Description</th>
</tr>
</thead>
</table>
| Native         | Select this option if the file contains data in the native format of the system where the generated programs will run.  
For mainframe, the native format is EBCDIC.  
For midrange or PC platforms, the native format is ASCII. |
| ASCII          | Select this option if the file contains data in ASCII format. |
| EBCDIC         | Select this option if the file contains data in EBCDIC format. |

**Note:** The Recording Mode for Standard Files with subtype Function will be disabled.

**Record Format**

Select the required Record Format from the drop-down list.

Default value = **Fixed**

**Note:** The Record Format for Standard Files with subtype Delimited will be disabled, and is always set to Variable.
Label

Select the required option from the drop-down list.
Default value = Standard

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Select this option if the file has standard label records (as defined by IBM). A label contains information such as the creation date and the length of the file.</td>
</tr>
<tr>
<td>Omit</td>
<td>Select this option if the file does not contain label records.</td>
</tr>
</tbody>
</table>

File Description Size

Mandatory field.
Enter the record size as a number of characters. The exact meaning depends on the selected Record Format (page 113).

Auto Calculate
If this option is activated, the size will be calculated automatically.

Block Size

Optional field. If no value is defined, the block size defaults to the record size.
Define the maximum number of characters contained in a block of records in the file. For variable-length records, the block size includes the four-byte record descriptor word for each record.
The block size is not allowed if the Undefined option is specified for the Record Format.
Auto Calculate
If this option is activated, the size will be calculated automatically.

Column Separator
This field contains the default column separator for Standard Files, if defined on the MetaMap Manager Settings screen of the MetaSuite INI Manager. For more information, refer to the Installation and Setup Guide. You can enter another character in this field in order to override the default column separator for this file.

Row Terminator
This field contains the default row terminator for Standard Files, if defined on the MetaMap Manager Settings screen of the MetaSuite INI Manager. For more information, refer to the Installation and Setup Guide. You can enter another character (sequence) in the field in order to override the default row terminator for this file.

External Source
Optional field.
You can use it in the two following situations:

1. You want to define a link between the Name definition in MetaSuite and the physical file name on disk. The physical file name will be automatically used in the generated job.
2. You want to indicate that there is no direct access to the data source. The data source might be remote, or there is need to copy or transform the source file via a simple copy or via a transformation tool. In this case the file, specified by the External Source parameter is the file that has to be transferred or copied to the local area. The local input file will have the name indicated by Name.

This parameter will be processed in the customizable MRL Tables. By default, MetaSuite will use the following conversion rules:

<table>
<thead>
<tr>
<th>External Source Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STD:Filename</td>
<td>This file name will be taken as input file. (This is the first method of using External Source)</td>
</tr>
<tr>
<td>FTP:Filename</td>
<td>The file will be downloaded via the FTP protocol.</td>
</tr>
<tr>
<td>CPY:Filename</td>
<td>A file copy will be done from this file to the standard file.</td>
</tr>
</tbody>
</table>

Note: These options are supported on the Windows and UNIX platform. Please contact the MetaSuite support team if you want to use this option on another platform.

Spanned
Select this check box if the variable-length records may "span" two or more blocks.
This option is only applicable if the Record Format is set to *Variable* the Block Size is smaller than the Record Size, and the Subtype is *Index*, *Relative* or *Sequential*.

**File Keys**

Optional field.

This field is available for Standard Files with subtype *Record Sequential*, *Index* or *VSAM*, and for SQL Table Groups.

**Note:** You can only define File Keys after having defined Data Structures (Standard Records or SQL Tables) and Data Entities (Standard Fields or SQL Columns).

1. Double-click the *New* icon available in the *File Keys* panel.
   A screen similar to this one is displayed:

2. First select the *Item* (the Record) using the drop-down list.
   All available fields for the selected record will be displayed underneath.

3. Select the required Standard Field and click *OK*.
   The *Standard File Key* properties panel is displayed.
4. Fill out the required fields.

   Name  By default, this field contains the name of the Data Entity you selected. If required you can modify the name.

   Field  Displays the name of the Field you selected.
   Click the button to display its properties.
   Click the button to remove the Field and select another one.

5. Apply your changes and close the File Key Properties panel. The new File Key is added to the File Keys panel.

6. Repeat this action for all File Keys you want to define.

7. If you need to change the order of the File Keys:
   • Right-click the key to be moved.
   • Select Move Forward or Move Backward from the pop-up menu, until the Key reaches its required position.

   **Note:** You can also create and delete File Keys using this pop-up menu.

**Business Tab (Standard Files)**

The fields on the Business Tab are identical for all Standard File data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Standard File Data Objects) (page 133) for a description of the fields.

**13.2. Defining Records**

1. In the Tree View Window, right-click the Standard File you want to define Records for and select Add Standard Record.

   The properties panel is displayed in the Workspace.
Two tabs are available: Technical and Business.

2. Fill out the required fields.
For a detailed description of the fields, refer to the sections:
- Technical Tab (Standard Records) (page 118)
- Business Tab (Standard File Data Objects) (page 133)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
In the Tree View Window, right-click the Standard File the new Record belongs to and select Save to MetaStore.

Note: If you do not add the Record to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

Technical Tab (Standard Records)
The following fields are available on the Technical tab:
- Name (page 118)
- Database (page 118)
- Size (page 119)
- CCSID (page 119)
- Record Keys (page 119)

Name
Mandatory field.
The name you enter must meet the following conditions:
- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

Database

Note: This field is specific for each file (subtype). Refer to the appropriate File Access Guide for more detailed information.

This field is mandatory for IMS Segments. Enter the segment name of the record, as specified in the DBDGEN and PSBGEN statements.
For RDBMS, the Creator Name should be entered in this field.
Size

Mandatory field.
Enter the maximum number of characters included in the record. The value must be an integer and cannot exceed the maximum record size defined for the Dictionary File it belongs to.

**Note:** If the size is changed manually and it is too small to contain all defined fields, it will be reset by the MetaStore Manager to the smallest size needed to contain all defined fields.

**Auto Calculate**

If this option is activated, the size will be calculated automatically.

**CCSID**

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.

**Record Keys**

Optional field.
This field only applies to fields containing multiple record types.
This option is used to identify Record identification fields and the specific ranges of values for those key fields. Record identification fields are used to identify the record type being defined.

1. **Double-click the New icon available in the Record Keys panel.**

**Note:** You can only define Record Keys after having defined Records and Fields for the Dictionary File.

The following screen is displayed:
2. Fill out the name of the new Key.

3. Select the required Field.
   Click the *Browse* button to display the list of all available Fields.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the *Select Item* drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

4. Select the Operator from the drop-down list.
   The following options work with the *Value* field only:
   - *Equal*
   - *Greater or equal*
   - *Greater*
   - *Less or equal*
   - *Less*
   - *Not equal*
   The following of the available options work with both the *Value* and the *High Value* fields:
   - *In range*
   - *Not in range*

5. Enter the Value.
   If you selected an operator working with the Value field only, enter the required value.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Value entered</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Not) Equal</td>
<td>3</td>
<td>Only records where the selected field has (not) value 3 are taken into account.</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Only records where the selected field has (not) value C are taken into account.</td>
</tr>
<tr>
<td>Greater or Equal</td>
<td>3</td>
<td>Only records where the selected field has a value greater than or equal to 3 are taken into account.</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Only records where the selected field has a value greater than or equal to C are taken into account.</td>
</tr>
<tr>
<td>Greater</td>
<td>3</td>
<td>Only records where the selected field has a value greater than 3 are taken into account.</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>Only records where the selected field has a value greater than C are taken into account.</td>
</tr>
</tbody>
</table>

Note: If you want to specify more than 1 value (or range of values), you have to define separate record keys.
If you selected an operator working with both the Value and High Value fields, enter the required value in these fields.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Value entered</th>
<th>High Value Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less or Equal</td>
<td>3</td>
<td>12</td>
<td>Only records where the selected field has a value lesser than or equal to 3 are taken into account.</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>Only records where the selected field has a value lesser than or equal to C are taken into account.</td>
</tr>
<tr>
<td>Less</td>
<td>3</td>
<td></td>
<td>Only records where the selected field has a value lesser than 3 are taken into account.</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>Only records where the selected field has a value lesser than C are taken into account.</td>
</tr>
</tbody>
</table>

6. Apply your changes and close the properties panel.
   The new Record Key is added to the Record Keys panel.

7. Repeat this action for all Record Keys you want to define.

**Business Tab (Standard Records)**

The fields on the Business Tab are identical for all Standard File data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Standard File Data Objects) (page 133) for a description of the fields.
13.3. Defining Fields

1. In the Tree View Window, right-click the Standard Record you want to define Fields for and select Add Standard Field.

   The properties panel is displayed in the Workspace.

   Two tabs are available: Technical and Business.

2. Fill out the required fields.

   For a detailed description of the fields, refer to the sections:
   • Technical Tab (Standard Fields) (page 123)
   • Business Tab (Standard File Data Objects) (page 133)

   Note: Some of the values can also be modified by double-clicking or selecting them from a drop-down list in the Record Fields Window.

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.

   In the Tree View Window, right-click the Standard File the new Field belongs to and select Save to MetaStore.

   Note: If you do not add the Field to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.
Technical Tab (Standard Fields)

The following fields are available on the Technical tab:

- **Properties**
  - Name (page 123)
  - Size (page 123)
  - Abs. Position (page 125)
  - Rel. Position (page 125)
  - Occurs (page 125)
  - Occurrence Depending On (page 125)

- **Content**
  - Type (page 126)
  - Decimals (page 127)
  - Unsigned (page 127)
  - Separated and Leading (page 128)
  - Initial (page 128)
  - Null (page 128)
  - Code (page 129)
  - Edit Mask (page 129)
  - Date Format (page 132)
  - Low Limit (page 132)
  - High Limit (page 132)
  - CCSID (page 132)

**Name**

Mandatory field.
The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Size**

In the Size field, enter an integer indicating the field length as a number of bytes. If the field occurs more than once in the record, this field indicates the length of a single occurrence of the field.
<table>
<thead>
<tr>
<th>Type</th>
<th>Additional</th>
<th>Size in # Bytes</th>
<th>COBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td># characters</td>
<td></td>
<td>PIC A(c) DISPLAY</td>
</tr>
<tr>
<td>Character</td>
<td># characters</td>
<td></td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>National</td>
<td>2 x # characters</td>
<td></td>
<td>PIC N(c)</td>
</tr>
<tr>
<td>Varchar</td>
<td># characters</td>
<td></td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>Binary</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td>Binary Native</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td>Bit</td>
<td></td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Decimal</td>
<td>Signed / Null-signed</td>
<td>(# digits + 1) / 2</td>
<td>PIC S9(n)V9(d) PACKED-DECIMAL</td>
</tr>
<tr>
<td>Float</td>
<td>8 digits of precision</td>
<td>4</td>
<td>COMP-1</td>
</tr>
<tr>
<td></td>
<td>17 digits of precision</td>
<td>8</td>
<td>COMP-2</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td></td>
<td>1 byte = 2 hexadecimal digits</td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>Numeric</td>
<td># digits</td>
<td></td>
<td>PIC S9(n)V9(d) DISPLAY</td>
</tr>
<tr>
<td>Printed Numeric</td>
<td># characters in Edit Mask</td>
<td></td>
<td>PIC EditMask DISPLAY</td>
</tr>
<tr>
<td>Printed Numeric National</td>
<td>2 x # characters in Edit Mask</td>
<td></td>
<td>PIC EditMask USAGE NATIONAL</td>
</tr>
</tbody>
</table>

Legend:
- $c$ = number of characters
- $n$ = number of integer digits
- $d$ = number of decimal digits
- $s$ = only present when value is signed. This option is not allowed for National.

**Note:** The Record Layout window shows the default value for the Field within the selected Record. The default value matches the size of the Field, as defined in the Record Fields Window.

**Auto Calculate**
If this option is activated, the size will be calculated automatically.
Abs. Position

In this field, you may change the Field's starting character position that was calculated automatically by the system. The value entered must be an integer.

**Note:** If the Record’s length is variable, you may not include the four-character record descriptor word when determining the start position of the Field.

Rel. Position

This is the 1-based position of this field within its domed structure (i.e., Record or Group).

- For a field that is not a subfield, the absolute and relative positions are equal.
- For a field \( x \) that is a subfield of group \( g \), the relative position will be calculated as \( 1 + \) (absolute position of \( x \)) - (absolute position of \( g \)).
  
  For example: if group \( g \) starts on position 5, and the absolute position of \( x \) is also 5, then the relative position of \( x \) is \( 1 + 5 - 5 = 1 \). In human terms: "\( x \) will start on the 1st position of \( g \)."

Auto Calculate

If this option is activated, the size will be calculated automatically.

Occurs

Optional field.

Enter an integer indicating the maximum number of times this field can occur in the record. If you do not define a specific value, the field is assumed to occur once.

Occurrence Depending On

Optional field. Only applies for repeating fields (n).
Select a numeric field that is defined prior to this field within the record.

Results:

- The value you defined with the **Occurs** option represents the *maximum* number of times this field can occur
- The value available in the selected field represents the *actual* number of times the field occurs.

1. Click the **Browse** button next to the **Occurrence Depending On** text field.
   
The list of available numeric fields is displayed.

2. Select the required Field.
   
The name of the selected Field is displayed in the **Occurrence Depending On** text field.

**Note:** If you click the Browse button again now, the Properties screen for the selected field will be displayed. If you want to select another field, you first have to delete the current field name from the text field and then click the Browse button.
Type

This field is mandatory and contains the default value Character.
There are two general classes of data types: non-numeric and numeric.

Non-numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td>This datatype indicates a field containing strings using the characters A-Z and a-z.</td>
</tr>
<tr>
<td>Character</td>
<td>This datatype indicates that the field may contain any possible character within the character set being used (including unprintable characters). It is not possible to perform numeric operations on a Character type field, even when the field contains only digits. Matching RDBMS Data Types: CHAR, BYTE, VARCHAR, VARCHAR2, LONG, RAW, LONGVARCHAR</td>
</tr>
<tr>
<td>Graphic</td>
<td>DB2 terminology for Character datatype.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>This datatype indicates that any hexadecimal characters are allowed within the field.</td>
</tr>
<tr>
<td>Long Varchar</td>
<td>This datatype indicates a field containing a variable character string of maximum 2 GB.</td>
</tr>
<tr>
<td>National</td>
<td>The National datatype is a subset of Unicode. The character set on which it is based is UTF-16, but restricted to two bytes per character.</td>
</tr>
<tr>
<td>Varchar</td>
<td>This datatype indicates a character field allowing the storage of values you need plus one byte for the length.</td>
</tr>
<tr>
<td>Vargraphic</td>
<td>DB2 terminology for Varchar datatype.</td>
</tr>
</tbody>
</table>

Numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>This datatype indicates that the field contains a binary numeric value composed of 0’s and 1’s.</td>
</tr>
<tr>
<td>Binary Native</td>
<td>This datatype indicates that the field contains binary native numbers. This means that its internal representation depends on the Operating System and/or processor.</td>
</tr>
<tr>
<td>Bit</td>
<td>This datatype indicates a field occupying a single bit storage. A BIT type field may only contain the binary values 0 or 1. It is possible to perform numeric operations on a BIT type field.</td>
</tr>
<tr>
<td>Byte</td>
<td>This datatype indicates a field occupying a single byte storage. A byte is 8 bits. It is possible to perform numeric operations on a BYTE type field.</td>
</tr>
<tr>
<td>Decimal</td>
<td>This datatype indicates a field containing a packed decimal value. Packed decimal is the most commonly used internal numeric data type. Two decimal digits are contained in each byte of a packed decimal number. However, if the number is signed, the last half byte contains a positive or negative sign indicator.</td>
</tr>
</tbody>
</table>
Decimals

If the field type allows the definition of decimals, this field contains the number of decimals. Decimals can be entered for the following field types:

- Binary
- Binary Native
- Decimal
- Numeric
- Printed Numeric
- Printed Numeric National

If the field type does not allow the definition of decimals, this field contains the default value 0.

Unsigned

Select the Unsigned check box to indicate that a numeric value is not signed, i.e. that it does not contain a sign indicator (+ or -).

This field applies for the following field types:

- Binary
- Binary Native
- Numeric

**Note:** For Decimal fields, the distinction between unsigned and positive signed INPUT fields is not relevant to the system.
Separated and Leading

These two check boxes allow to define how over-punched and separate sign indicators must be treated. They only apply for the Numeric field type.

There are several possibilities.
If the sign indicator is over-punched in the number, select the Unsigned check box above.

- Select Leading to put the sign on the initial digit.
- Select Separate to put the sign on the last byte (no digit).
- Select Leading and Separate to put the sign in the initial byte (no digit).

If the Numeric field contains a separate plus or minus sign attached to the number, select the Separate check box.

Initial

Optional field.

Enter the initial value for the field.

This value is only taken into account for Target Files. If an initial value was defined for a Source file, this parameter will not affect any operation, except for syntax checking.

For Numeric fields, this is a numeric constant, where the decimal point should be given by a . (point).

For Character fields, this may be an alphanumeric constant (not enclosed by quotes) or the system fields SYS-LOW-VALUE or SYS-HIGH-VALUE, which correspond to system fields LOW-VALUE and HIGH-VALUE within COBOL.

Null

This field indicates the nullable status of the field, and whether Inbound or Outbound nulls are used.

**Note:** If this field is left blank, the default NULLABLE value defined in the MetaSuite dictionary will be applied to this field. For more information, refer to the section Create Dictionary/Enter License Key in the Generator Manager Guide.

Select the required null-indicator from the drop-down list.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Select this option if no nullability information is known about the field, or when it is of no importance for the field.</td>
</tr>
<tr>
<td>Default</td>
<td>Select this option if this field is a NotNull field with a default value. The Default notion is only documentary metadata for a field.</td>
</tr>
<tr>
<td></td>
<td>The default Nullable value defined in the MetaSuite dictionary will be applied. Refer to the section Create a Dictionary/Enter License Key in the Generator Manager User Guide.</td>
</tr>
<tr>
<td>NotNull</td>
<td>Select this option if a Null Value should never be allowed in this field.</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
Code | This field is treated as a code, not as a number.
No Code | Resets the code operator to zero. Select this option when there is no additional information to be added for the field.
Time | Select this option to define a field that contains TIME information.
Timestamp | Select this option to define a field that contains TIMESTAMP information.

Edit Mask

This field is mandatory for Printed Numeric fields, but optional for other field types. It indicates how the alphanumeric values must be formatted.

*Note:* When the Edit Mask is set for a printed numeric field, the MetaStore Manager will reset the size of the field to the size that corresponds to the chosen Edit Mask.
Enter the characters defining a Mask in this text field. This Mask will override the default mask for the field. There is a default mask for each field type. Both the default masks and the manually created masks are composed of Replacement and Insertion characters.

The following table lists the Replacement characters and their meaning. Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field.

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>N</td>
<td>National (2-byte Unicode character set)</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
</tbody>
</table>

The following table lists the Insertion characters and their meaning. Insertion characters indicate characters to be printed in addition to those contained in the stored field.

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
<tr>
<td>,</td>
<td>Comma (separator for the sake of large number readability or decimal separator depending on the “Decimal Separator” option in the INI file)</td>
</tr>
<tr>
<td>.</td>
<td>Decimal point (separator for the sake of large number readability or decimal separator, depending on the “Decimal Separator” option in the INI file)</td>
</tr>
<tr>
<td>B</td>
<td>Blank</td>
</tr>
<tr>
<td>-</td>
<td>Floating or trailing minus for negative values. Plus is not accepted as valid character. Positive numbers will have a blank as sign character.</td>
</tr>
<tr>
<td>+</td>
<td>Floating or trailing plus or minus sign</td>
</tr>
<tr>
<td>CR</td>
<td>Trailing credit symbol for negative values only</td>
</tr>
<tr>
<td>DB</td>
<td>Trailing debit symbol for negative values</td>
</tr>
<tr>
<td>V</td>
<td>Virtual comma</td>
</tr>
</tbody>
</table>

As mentioned above, there is a default Mask for each field type:
Examples of default masks:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Size</th>
<th>Default mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields</td>
<td>6</td>
<td>999999-</td>
</tr>
<tr>
<td>Signed numeric fields with two decimals</td>
<td>6</td>
<td>99999.99-</td>
</tr>
<tr>
<td>Character Field</td>
<td>6</td>
<td>X(6)</td>
</tr>
<tr>
<td>Unicode Field</td>
<td>6</td>
<td>N(3)</td>
</tr>
</tbody>
</table>

You may also define customized masks.
Examples:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>6</td>
<td>XXBXXXX</td>
<td>AB138B</td>
<td>AB 138B</td>
</tr>
<tr>
<td>Numeric with 2 decimal positions</td>
<td>2</td>
<td>.99</td>
<td>.350</td>
<td>.350</td>
</tr>
<tr>
<td>Numeric with 3 minus signs</td>
<td>4</td>
<td>--9</td>
<td>0 -1 210</td>
<td>0 -1 210</td>
</tr>
<tr>
<td>Numeric with 3 plus signs</td>
<td>4</td>
<td>+++9</td>
<td>0 -1 210</td>
<td>+0 -1 210</td>
</tr>
</tbody>
</table>
**Date Format**

If the field must contain a date, select the required date format from the drop-down list. The system automatically validates date fields whenever they are referenced in a MetaMap model, and automatically converts date fields whenever they are compared to another date or used in a calculation. In all the available formats, YY or YYYY stands for the year and MM stands for the month. DD stands for the day within a month and DDD stands for the day within the year.

When the format contains a '?', this indicates the date delimiter that is used. Data formats with a '?' are only supported for Character and Varchar field types. When the data format does not contain a '?', the different parts in the date are not delimited by a special character.

The Date Format list is accessible for the following field types:

- Binary
- Binary Native
- Character
- Decimal
- Numeric
- National
- Varchar

**Note:** When a date format is chosen, MetaStore Manager will reset the size of the field to the size that corresponds to the chosen date format.

**Low Limit**

Optional field.

When a *Low Limit* is specified, you must define a value in the *High Limit* field as well.

You may use this field to define a minimum value for this field. If a lower value is encountered, the system will consider the data invalid.

**High Limit**

Optional field.

When a *High Limit* is specified, you must define a value in the *Low Limit* field as well. You may use this field to define a maximum value for this field. If a higher value is encountered, the system will consider the data invalid.

**CCSID**

Enter the Coded Character Set Identifier.

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.

**Business Tab (Standard Fields)**

The fields on the Business Tab are identical for all Standard File data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Standard File Data Objects) (page 133) for a description of the fields.

### 13.4. Business Tab (Standard File Data Objects)

The following fields are available on the Business tab:

- **Business Rule** (page 133)
- **Note** (page 133)

**Note:** If you want to enter text in RTF (Rich Text Format), right-click and select RTF from the context menu (or use the shortcut CTRL + R).

**Business Rule**

Optional field.

Enter a free-form description for the data object (Dictionary File, Record, Field, Relationship or Index). For example, the business rule or the requirements.

**Note**

Optional field.

Enter a free-form note for the data object.
In an IDMS Subschema, the data are stored in one centralized location beyond the scope of the defined application program. An IDMS record is a collection of related data items or fields and can belong to many Sets. Each Set defines a logical relationship between record types.

The following table shows how the IDMS terminology can be matched with the standard MetaSuite terminology.

<table>
<thead>
<tr>
<th>IDMS Terminology</th>
<th>MetaSuite Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema</td>
<td>File</td>
</tr>
<tr>
<td>Record Type</td>
<td>Record</td>
</tr>
<tr>
<td>Field</td>
<td>Field</td>
</tr>
<tr>
<td>Index</td>
<td>Index</td>
</tr>
<tr>
<td>Set</td>
<td>Link</td>
</tr>
</tbody>
</table>

This chapter explains the following actions:

- Creating the Dictionary File (page 135)
- Defining Records (page 137)
- Defining Fields (page 140)
- Defining Relationships (page 151)
- Defining Indices (page 155)

For more technical information, refer to the IDMS File Access Guide.

1. In the Tree View Window, right-click the MetaStore root icon and select Add > Subschema. The properties panel appears in the Workspace.

   ![IDMS Subschema](image)

   Two tabs are available: Technical and Business.

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Subschemas) (page 135)
   - Business Tab (Subschema Data Objects) (page 158)

3. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the new Dictionary File and select Save to MetaStore.

   **Note:** If you do not save the Dictionary File to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Subschemas)**

The following fields are available on the Technical tab:

- **Name** (page 135)
- **Code-control** (page 136)
- **Database** (page 136)
- **Version** (page 136)
- **CCSID** (page 136)
- **Subtype** (page 137)
- **Schema** (page 137)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
• It must begin with an alphabetic character.
• It may contain the characters a-z, A-Z and 0-9.
• It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
• It may not be ended with a – (hyphen).
• File names must be unique in the MetaStore Repository.

**Code-control**

Optional field.

Code-control Tables contain information about how to process a certain type of data container. Each table element of the code-control table points to a code table in the COBOL Generator Dictionary. Each data container type has a pre-defined code-control table. If required, this default code-control table can be overwritten. For example: in order to invoke an external I/O module to read or write the file or to expand or compress data.

**Note:** When the option is used, the selected table must exist in the COBOL Generator Dictionary before programs using the file can be generated.

For more information about code-control tables, refer to the section *Code-control Tables* in the *Generator Manager Guide*.

**Database**

Enter the name of the corresponding database, if applicable.

**Note:** This field is mandatory for Subschemas.

Enter the name of the default subschema and optionally the default CA-IDMS database in the following format: Subschema-name[.Database-name]

This name will be used by MetaMap when accessing the IDMS Subschema. The name must be enclosed in single quotes.

**Version**

Default value = 1

You can change the version number, if required.

When reimporting or recollecting an existing data container, the version will be modified depending on the settings specified in the INI Manager. For more information, refer to the chapter MetaStore Manager Settings in the *MetaSuite INI Manager Guide*.

**CCSID**

Enter the Coded Character Set Identifier.

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL. This CCSID is an enriched property and will not be collected.

Subtype

Select the required subtype from the drop-down list. The following options are available:

- IDMS/R
- IDMS/X
- IDXII

Schema

Mandatory field. Enter the name of the CA-IDMS schema.

Business Tab (Subschemas)

The fields on the Business Tab are identical for all Subschema data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Subschema Data Objects) (page 158) for a description of the fields.

14.2. Defining Records

1. In the Tree View Window, right-click the appropriate IDMS Subschema Dictionary File and select Add Subschema Record. The properties panel is displayed in the Workspace.

   ![Image of properties panel]

   This screen contains two tabs: Technical and Business.

2. Fill out the required fields.
For a detailed description of the fields, refer to the sections:

- Technical Tab (Subschema Records) (page 138)
- Business Tab (Subschema Data Objects) (page 158)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Dictionary File the new Record belongs to and select Save to MetaStore.

**Note:** If you do not add the File to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

### Technical Tab (Subschema Records)

The following fields are available on the Technical tab:

- **Name** (page 138)
- **Size** (page 138)
- **Database** (page 139)
- **Area Name** (page 139)
- **Subschema Record Key** (page 139)
- **CCSID** (page 139)

#### Name

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

#### Size

Mandatory field.

Enter the record size. The value must be at least as large as the record LENGTH shown in the IDMSRPTS Subschema Data Dictionary Listing.

**Auto Calculate**

If this option is activated, the size will be calculated automatically.
Database

Enter *BDAM* in this field, if the field must be considered as a "dummy" field describing the record identifier portion of a BDAM actual key.

Area Name

Mandatory field.

IDMS organizes its databases as a series of files. These files are mapped and pre-formatted into so-called areas. The areas are subdivided into pages which correspond to physical blocks on the disk. The database records are stored within these blocks. The DBA allocates a fixed number of pages in a file for each area. The DBA then defines which records are to be stored in each area, and details of how they are to be stored.

Enter the WITHIN name as it appears on the IDMSRPTS Subschema Record Description Listing for the record being defined.

Subschema Record Key

This record key is also known as the storage key for the CALC method. Four methods are available for storing records in an IDMS database: Direct, Sequential, CALC, and VIA. CALC uses a hashing algorithm to decide where to place the record; the hash key then provides efficient retrieval of the record. The entire CALC area is preformatted each with a header consisting of a special CALC "owner" record. The hashing algorithm determines a page number (from which the physical disk address can be determined), and the record is then stored on this page, or as near as possible to it, and is linked to the header record on that page using the CALC set. The CALC records are linked to the page's CALC Owner record using a single link-list (pointers). The CALC Owner located in the page header thus owns the set of all records which target to its particular page (whether the records are stored on that page or, in the case of an overflow, on another page).

CALC provides extremely efficient storage and retrieval: IDMS can retrieve a CALC record in 1.1 I/O operations. However, the method does not cope well with changes to the value of the primary key, and expensive reorganization is needed if the number of pages needs to be expanded.

Select the name of the field, by which you can randomly obtain the record.

1. Click the *Browse* button to display the Subschema Record Key Properties panel.

2. Fill out the name of the new Subschema Record Key.

3. Select the required Subschema Field.
   Click the *Browse* button to display the list of all available fields and subfields.

4. Apply your changes and close the Subschema Record Key Properties panel.
   The new Record Key is displayed in the Subschema Record Key field.

CCSID

Enter the Coded Character Set Identifier.

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL. This CCSID is an enriched property and will not be collected.

**Business Tab (Subschema Records)**

The fields on the Business Tab are identical for all Subschema data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section [Business Tab (Subschema Data Objects)](page 158) for a description of the fields.

### 14.3. Defining Fields

1. In the Tree View Window, right-click the IDMS Subschema Record you want to define Elements for and select **Add Subschema Element**. The properties panel is displayed in the Workspace.

   ![Screenshot of the properties panel](image.png)

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - [Technical Tab (Subschema Elements)](page 141)
   - [Business Tab (Subschema Data Objects)](page 158)

**Note:** Some of the values can also be modified by double-clicking or selecting them from a drop-down list in the Record Fields Window.
3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Subschema the new Element belongs to and select `Save to MetaStore`.

   **Note:** If you do not add the Element to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Subschema Elements)**

The following fields are available on the Technical tab:

- **Properties**
  - **Name** (page 141)
  - **Size** (page 142)
  - **Abs. Position** (page 143)
  - **Rel. Position** (page 143)
  - **Occurs** (page 143)
  - **Occurrence Depending On** (page 143)

- **Content**
  - **Type** (page 144)
  - **Decimals** (page 145)
  - **Unsigned** (page 145)
  - **Separated and Leading** (page 146)
  - **Initial** (page 146)
  - **Null** (page 146)
  - **Code** (page 147)
  - **Edit Mask** (page 148)
  - **Date Format** (page 150)
  - **Low Limit** (page 150)
  - **High Limit** (page 150)
  - **CCSID** (page 151)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
• It may not be ended with a – (hyphen).
• File names must be unique in the MetaStore Repository.

Size
In the Size field, enter an integer indicating the field length as a number of bytes. If the field occurs more than once in the record, this field indicates the length of a single occurrence of the field.

<table>
<thead>
<tr>
<th>Type</th>
<th>Additional</th>
<th>Size in # Bytes</th>
<th>COBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td># characters</td>
<td>PIC A(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Character</td>
<td># characters</td>
<td>PIC X(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>National</td>
<td>2 x # characters</td>
<td>PIC N(c)</td>
<td></td>
</tr>
<tr>
<td>Varchar</td>
<td># characters</td>
<td>PIC X(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Binary</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td>Binary Native</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td>Bit</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decimal</td>
<td>Signed / Null-signed</td>
<td>(# digits + 1) / 2</td>
<td>PIC S9(n)V9(d) PACKED-DECIMAL</td>
</tr>
<tr>
<td>Float</td>
<td>8 digits of precision</td>
<td>4</td>
<td>COMP-1</td>
</tr>
<tr>
<td></td>
<td>17 digits of precision</td>
<td>8</td>
<td>COMP-2</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>1 byte = 2 hexadecimal digits</td>
<td></td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>Numeric</td>
<td># digits</td>
<td>PIC S9(n)V9(d) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Printed Numeric</td>
<td># characters in Edit Mask</td>
<td>PIC EditMask DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Printed Numeric National</td>
<td>2 x # characters in Edit Mask</td>
<td>PIC EditMask USAGE NATIONAL</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
• $c$ = number of characters
• $n$ = number of integer digits
• $d$ = number of decimal digits
• $s$ = only present when value is signed. This option is not allowed for National.

Note: The Record Layout window shows the default value for the Field within the selected Record. The default value matches the size of the Field, as defined in the Record Fields Window.
Auto Calculate
If this option is activated, the size will be calculated automatically.

Abs. Position
In this field, you may change the Field's starting character position that was calculated automatically by the system. The value entered must be an integer.

Note: If the Record's length is variable, you may not include the four-character record descriptor word when determining the start position of the Field.

Rel. Position
This is the 1-based position of this field within its domed structure (i.e., Record or Group).
- For a field that is not a subfield, the absolute and relative positions are equal.
- For a field $x$ that is a subfield of group $g$, the relative position will be calculated as $1 + (\text{absolute position of } x) - (\text{absolute position of } g)$.
  
  For example: if group $g$ starts on position 5, and the absolute position of $x$ is also 5, then the relative position of $x$ is $1+5-5 = 1$. In human terms: "$x$ will start on the 1st position of $g$".

Auto Calculate
If this option is activated, the size will be calculated automatically.

Occurs
Optional field.
Enter an integer indicating the maximum number of times this field can occur in the record. If you do not define a specific value, the field is assumed to occur once.

Occurrence Depending On
Optional field. Only applies for repeating fields ($n$).
Select a numeric field that is defined prior to this field within the record.

Results:
- The value you defined with the Occurs option represents the maximum number of times this field can occur.
- The value available in the selected field represents the actual number of times the field occurs.

1. Click the Browse button next to the Occurrence Depending On text field.
The list of available numeric fields is displayed.
2. Select the required Field.
   The name of the selected Field is displayed in the *Occurrence Depending On* text field.

   **Note:** If you click the Browse button again now, the Properties screen for the selected field will be displayed. If you want to select another field, you first have to delete the current field name from the text field and then click the Browse button.

### Type

This field is mandatory and contains the default value *Character*.

There are two general classes of data types: *non-numeric* and *numeric*.

#### Non-numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td>This datatype indicates a field containing strings using the characters A-Z and a-z.</td>
</tr>
<tr>
<td>Character</td>
<td>This datatype indicates that the field may contain any possible character within the character set being used (including <em>unprintable</em> characters). It is not possible to perform numeric operations on a Character type field, even when the field contains only digits. Matching RDBMS Data Types: CHAR, BYTE, VARCHAR, VARCHAR2, LONG, RAW, LONGVARCHAR</td>
</tr>
<tr>
<td>Graphic</td>
<td>DB2 terminology for Character datatype.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>This datatype indicates that any hexadecimal characters are allowed within the field.</td>
</tr>
<tr>
<td>Long Varchar</td>
<td>This datatype indicates a field containing a variable character string of maximum 2 GB.</td>
</tr>
<tr>
<td>National</td>
<td>The National datatype is a subset of Unicode. The character set on which it is based is UTF-16, but restricted to two bytes per character.</td>
</tr>
<tr>
<td>Varchar</td>
<td>This datatype indicates a character field allowing the storage of values you need plus one byte for the length.</td>
</tr>
<tr>
<td>Vargraphic</td>
<td>DB2 terminology for Varchar datatype.</td>
</tr>
</tbody>
</table>

#### Numeric Data Types

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>This datatype indicates that the field contains a binary numeric value composed of 0's and 1's.</td>
</tr>
<tr>
<td>Binary Native</td>
<td>This datatype indicates that the field contains binary native numbers. This means that its internal representation depends on the Operating System and/or processor.</td>
</tr>
<tr>
<td>Bit</td>
<td>This datatype indicates a field occupying a single bit storage. A BIT type field may only contain the binary values 0 or 1. It is possible to perform numeric operations on a BIT type field.</td>
</tr>
</tbody>
</table>
Decimals

If the field type allows the definition of decimals, this field contains the number of decimals. Decimals can be entered for the following field types:

- Binary
- Binary Native
- Decimal
- Numeric
- Printed Numeric
- Printed Numeric National

If the field type does not allow the definition of decimals, this field contains the default value 0.

Unsigned

Select the Unsigned check box to indicate that a numeric value is not signed, i.e. that it does not contain a sign indicator (+ or -).

This field applies for the following field types:

- Binary
- Binary Native
• Numeric

**Note:** For Decimal fields, the distinction between unsigned and positive signed INPUT fields is not relevant to the system.

**Separated and Leading**

These two check boxes allow to define how over-punched and separate sign indicators must be treated. They only apply for the Numeric field type.

There are several possibilities.

If the sign indicator is over-punched in the number, select the *Unsigned* check box above.

- Select *Leading* to put the sign on the initial digit.
- Select *Separate* to put the sign on the last byte (no digit).
- Select *Leading* and *Separate* to put the sign in the initial byte (no digit).

If the Numeric field contains a separate plus or minus sign attached to the number, select the *Separate* check box.

**Initial**

Optional field.

Enter the initial value for the field.

This value is only taken into account for Target Files. If an initial value was defined for a Source file, this parameter will not affect any operation, except for syntax checking.

For Numeric fields, this is a numeric constant, where the decimal point should be given by a . (point).

For Character fields, this may be an alphanumeric constant (not enclosed by quotes) or the system fields `SYS-LOW-VALUE` or `SYS-HIGH-VALUE`, which correspond to system fields `LOW-VALUE` and `HIGH-VALUE` within COBOL.

**Null**

This field indicates the nullable status of the field, and whether Inbound or Outbound nulls are used.

**Note:** If this field is left blank, the default NULLABLE value defined in the MetaSuite dictionary will be applied to this field. For more information, refer to the section Create Dictionary/Enter License Key in the Generator Manager Guide.

Select the required null-indicator from the drop-down list.
<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Select this option if this field is a NotNull field with a default value. The Default notion is only documentary metadata for a field. The default Nullable value defined in the MetaSuite dictionary will be applied. Refer to the section Create a Dictionary/Enter License Key in the Generator Manager User Guide.</td>
</tr>
<tr>
<td>NotNull</td>
<td>Select this option if a Null Value should never be allowed in this field.</td>
</tr>
</tbody>
</table>
| InNull  | Select this option if a Null Value is allowed in this field. The used value to define a Null value is determined by a default setting in the MetaSuite COBOL Generator. When a Null Value is assigned, the first position or character of the field itself indicates the Null Value (the so called inbound Null).  
**Note:** In case of National or PRN-National, the Null Value is indicated by two bytes. |
| OutNull | Select this option if Null Values are allowed in this field. To store a Null value, an additional placeholder is foreseen in the sequential file that precedes the real field.  
When a Null value is assigned, this additional placeholder indicates the Null value (the so called left outbound Null). The used value to define a Null value is determined by a default setting in the MetaSuite COBOL Generator.  
**Note:** This placeholder contains one byte in order to store the Null indicator, except in case of field type National or PRN-National. In that case the placeholder contains two bytes. |
| OutNullR| Select this option if Null Values are allowed in this field. To store a Null value, an additional placeholder is foreseen in the sequential file that follows the real field.  
When a Null value is assigned, this additional placeholder indicates the Null value (the so called right outbound Null). The used value to define a Null value is determined by a default setting in the MetaSuite COBOL Generator.  
**Note:** This placeholder contains one byte in order to store the Null indicator, except in case of field type National or PRN-National. In that case the placeholder contains two bytes. |

### Code

Default value = *No Code*

The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>This field is treated as a code, not as a number.</td>
</tr>
<tr>
<td>No Code</td>
<td>Resets the code operator to zero. Select this option when there is no additional information to be added for the field.</td>
</tr>
<tr>
<td>Time</td>
<td>Select this option to define a field that contains TIME information.</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Select this option to define a field that contains TIMESTAMP information.</td>
</tr>
</tbody>
</table>
**Edit Mask**

This field is mandatory for Printed Numeric fields, but optional for other field types. It indicates how the alphanumeric values must be formatted.

**Note:** When the Edit Mask is set for a printed numeric field, the MetaStore Manager will reset the size of the field to the size that corresponds to the chosen Edit Mask.

Enter the characters defining a Mask in this text field. This Mask will override the default mask for the field. There is a default mask for each field type. Both the default masks and the manually created masks are composed of *Replacement* and *Insertion* characters.

The following table lists the Replacement characters and their meaning. Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field.

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>N</td>
<td>National (2-byte Unicode character set)</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
</tbody>
</table>

The following table lists the Insertion characters and their meaning. Insertion characters indicate characters to be printed in addition to those contained in the stored field.

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
<tr>
<td>,</td>
<td>Comma (separator for the sake of large number readability or decimal separator depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>.</td>
<td>Decimal point (separator for the sake of large number readability or decimal separator, depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>B</td>
<td>Blank</td>
</tr>
<tr>
<td>-</td>
<td>Floating or trailing minus for negative values. Plus is not accepted as valid character. Positive numbers will have a blank as sign character.</td>
</tr>
<tr>
<td>+</td>
<td>Floating or trailing plus or minus sign</td>
</tr>
<tr>
<td>CR</td>
<td>Trailing credit symbol for negative values only</td>
</tr>
</tbody>
</table>
As mentioned above, there is a default Mask for each field type:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Default Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields</td>
<td>The default mask contains a minus sign as the rightmost character. All negative values are printed with a trailing minus sign.</td>
</tr>
<tr>
<td>Numeric fields with decimals</td>
<td>The default mask contains a decimal point and as many digit replacement characters (9s) to the right of the decimal point as are specified by the Decimal option.</td>
</tr>
<tr>
<td>Numeric fields</td>
<td>The default mask contains as many digits as its size, without zero suppression.</td>
</tr>
<tr>
<td>Date fields</td>
<td>The default mask is its selected date format.</td>
</tr>
<tr>
<td>Alphanumeric fields</td>
<td>The default mask contains as many alphanumeric character replacement characters (X) as are required to print the field.</td>
</tr>
</tbody>
</table>

Examples of default masks:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Size</th>
<th>Default mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields without decimal positions</td>
<td>6</td>
<td>9999999-</td>
</tr>
<tr>
<td>Signed numeric fields with two decimals</td>
<td>6</td>
<td>9999.99-</td>
</tr>
<tr>
<td>Character Field</td>
<td>6</td>
<td>X(6)</td>
</tr>
<tr>
<td>Unicode Field</td>
<td>6</td>
<td>N(3)</td>
</tr>
</tbody>
</table>

You may also define customized masks.

Examples:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>6</td>
<td>XXBXXX</td>
<td>AB138B</td>
<td>AB 138B</td>
</tr>
<tr>
<td>Numeric with 2 decimal positions</td>
<td>2</td>
<td>.99</td>
<td>.35</td>
<td>.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-.12</td>
<td>.12</td>
</tr>
<tr>
<td>Numeric with 3 minus signs</td>
<td>4</td>
<td>---9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>
Date Format

If the field must contain a date, select the required date format from the drop-down list. The system automatically validates date fields whenever they are referenced in a MetaMap model, and automatically converts date fields whenever they are compared to another date or used in a calculation. In all the available formats, YY or YYYY stands for the year and MM stands for the month. DD stands for the day within a month and DDD stands for the day within the year.

When the format contains a ‘?’ , this indicates the date delimiter that is used. Data formats with a ‘?’ are only supported for Character and Varchar field types. When the data format does not contain a ‘?’ , the different parts in the date are not delimited by a special character.

The Date Format list is accessible for the following field types:

- Binary
- Binary Native
- Character
- Decimal
- Numeric
- National
- Varchar

**Note:** When a date format is chosen, MetaStore Manager will reset the size of the field to the size that corresponds to the chosen date format.

Low Limit

Optional field.

When a *Low Limit* is specified, you must define a value in the *High Limit* field as well.

You may use this field to define a minimum value for this field. If a lower value is encountered, the system will consider the data invalid.

High Limit

Optional field.

When a *High Limit* is specified, you must define a value in the *Low Limit* field as well. You may use this field to define a maximum value for this field. If a higher value is encountered, the system will consider the data invalid.
CCSID

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.

Business Tab (Subschema Elements)
The fields on the Business Tab are identical for all Subschema data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Subschema Data Objects) (page 158) for a description of the fields.

14.4. Defining Relationships
The purpose of defining Relationships is to define CA-IDMS sets to the MetaStore as link entities.

1. In the Tree View Window, right-click the new Subschema and select Add Subschema Relationship.
The properties panel is displayed in the Workspace.

Each LINK definition relates two MetaSuite (Subschema) records to one another. The "basis" of the link is a field in one record (the "from" record, so called because data is taken from that record and used to locate the other record) and a descriptor in the other (the "to" record, so called because the descriptor leads to that record).
Two tabs are available: Technical and Business.

2. Fill out the required fields.
For a detailed description of the fields, refer to the sections:

- Technical Tab (Subschema Relationships) (page 152)
- Business Tab (Subschema Data Objects) (page 158)

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Subschema File Group and select *Save to MetaStore*.

![Note: If you do not save the Relationship to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.](image)

**Technical Tab (Subschema Relationships)**

The following fields are available on the Technical tab:

- **Name** (page 152)
- **Database** (page 152)
- **Relationship From** (page 152)
- **Optional check box** (page 154)
- **Relationships To** (page 154)

**Name**

Mandatory field.

Enter the name of the Relationship as shown next to the SET statement in the Subschema Set Description Listing for the subschema.

**Database**

This is the unique name for the Database Entity (32 characters).

**Relationship From**

**Note:** Enter the name of the record type specified next to the OWNER statement in the Subschema Set Description Listing. If the name in the Subschema Set Description Listing is in the format IXOWNER, the set must be added as an Index to the MetaStore Repository.

Mandatory field.

For more technical information, refer to the section *ADD LINK* in the *IDMS File Access Guide*. 
1. Fill out the name of the new Relationship key.

2. Click the button next to the Relationship From text field. A screen similar to this one is displayed:

   ![](image)

3. First select the required Item (Subschema or Subschema Record) using the drop-down list. All available elements for the selected Subschema or Subschema Record will be displayed underneath. Two extra options are available at the top right of the pop-up window for selecting the required item:
   - **Show all**
     - When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - **Indentation**
     - When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

4. Select the required Subschema Elements and click OK. The Subschema Relationship From properties panel is displayed.

   ![](image)

5. Fill out the required fields.

<table>
<thead>
<tr>
<th>Name</th>
<th>The name of the new Relationship</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Record</td>
<td>In order to define a particular relationship between two records, the user has to specify the first record of the relationship in the From Record field.</td>
</tr>
</tbody>
</table>

**Note:** You can use the Browse button to display the list of available items.
6. Apply your changes and close the properties panel. The name will be displayed in the Relationship From field.

**Optional check box**

Select the Optional check box if the participation of a given record type in the set is optional, meaning that CA-IDMS allows the link between two record types to be either present or absent under user-defined conditions.

If the MEMBER line of the IDMSRPTS Subschema Set Description Listing for the set contains the word OPTIONAL, you must select this check box.

**Relationships To**

1. Double-click the New icon available in the Relationships To text zone.
   A screen similar to this one is displayed:

   ![Subschema Record](image)

2. First select the required File Group or File using the drop-down list.

3. Select the required Field and click OK.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.
   The Subschema Relationship To properties panel is displayed.
4. Fill out the required fields.
   For more technical information, refer to the section ADD LINK in the IDMS File Access Guide.
   
   **Name**  
   The name of the new Relationship

   **To Record**  
   In order to define a particular relationship between two records, the user has to specify the second record of the relationship in the To Record field.

   **Note:** You can use the Browse button to display the list of available items.

5. Apply your changes and close the properties panel.
   The relationship key name is added in the Relationships To panel.

**Business Tab (Subschema Relationships)**

The fields on the Business Tab are identical for all Subschema data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Subschema Data Objects) (page 158) for a description of the fields.

### 14.5. Defining Indices

An IDMS Subschema index declares the CA-IDMS to the MetaStore Repository.

1. In the Tree View Window, right-click the Subschema you want to define an Index for and select Add | Subschema Index.
   The properties panel is displayed in the Workspace:

   Two tabs are available: **Technical** and **Business**.

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Subschema Indices) (page 156)
   - Business Tab (Subschema Data Objects) (page 158)
3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Subschema the new Index belongs to and select Save to MetaStore.

   **Note:** If you do not add the Index to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

**Technical Tab (Subschema Indices)**

The following fields are available on the Technical tab:

- **Name** (page 156)
- **Database** (page 156)
- **Index Keys** (page 156)

**Name**

Mandatory field.
Enter the name of a CA-IDMS set as identified by SET statement in the Subschema Set Description Listing.

**Database**

Enter the name of the corresponding database.

**Index Keys**

Mandatory field.
Select the keyfield for the set, as identified by the MEMBER statement in the Subschema Set Description Listing.

**Note:** You can only define Index Keys after having saved the Index.
1. Double-click the New icon available in the Index Keys panel.
   The following screen is displayed.

   ![Image of Index Keys panel]

2. Select the required Field and click OK.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.
   The Subschema Index Key properties panel is displayed.

   ![Image of Subschema Index Key properties panel]

3. Fill out the required fields.

   **Name**
   The name of the new Index.
   The Index-set-name is the name of a CA-IDMS set (identified by SET in the Subschema Set Description Listing).

   **Field**
   The Index-field-name name of the keyfield for the set, as shown following the MEMBER record name in the Subschema Set Description Listing.

4. Apply your changes.
   The Index Key name is added in the Index Keys panel.

**Business Tab (Subschema Indices)**

The fields on the Business Tab are identical for all Subschema data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Subschema Data Objects) (page 158) for a description of the fields.
14.6. Business Tab (Subschema Data Objects)

The following fields are available on the Business tab:

- [Business Rule](page 158)
- [Note](page 158)

**Note:** If you want to enter text in RTF (Rich Text Format), right-click and select RTF from the context menu (or use the shortcut CTRL + R).

**Business Rule**

Optional field.
Enter a free-form description for the data object (Dictionary File, Record, Field, Relationship or Index). For example, the business rule or the requirements.

**Note**

Optional field.
Enter a free-form note for the data object.
This chapter explains the following actions:

- Creating the Dictionary File (page 159)
- Defining Records (page 161)
- Defining Fields (page 164)
- Defining Relationships (page 175)

### 15.1. Creating the Dictionary File

1. In the Tree View Window, right-click the MetaStore root icon and select Add > Supra Database. The properties panel is displayed in the Workspace.

![Properties panel for Supra Database](image)

Two tabs are available: Technical and Business.

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Supra Databases) (page 160)
   - Business Tab (Supra Data Objects) (page 179)

3. Apply or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the new Supra Database and select Save to MetaStore.

---

**Note:** If you do not save the Supra Database to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.
Technical Tab (Supra Databases)

The following fields are available on the Technical tab:

- Name (page 160)
- Code-control (page 160)
- Database (page 160)
- Version (page 160)
- CCSID (page 161)

Name

Mandatory field.
The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

Code-control

Optional field.
Code-control Tables contain information about how to process a certain type of data container. Each table element of the code-control table points to a code table in the COBOL Generator Dictionary.

Each data container type has a pre-defined code-control table. If required, this default code-control table can be overwritten. For example: in order to invoke an external I/O module to read or write the file or to expand or compress data.

Note: When the option is used, the selected table must exist in the COBOL Generator Dictionary before programs using the file can be generated.

For more information about code-control tables, refer to the section Code-control Tables in the Generator Manager Guide.

Database

Enter the name of the corresponding database, if applicable.

Version

Default value = 1
You can change the version number, if required.
When reimporting or recollecting an existing data container, the version will be modified depending on the settings specified in the INI Manager. For more information, refer to the chapter MetaStore Manager Settings in the *MetaSuite INI Manager Guide*.

**CCSID**

Enter the Coded Character Set Identifier.

CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.

This CCSID is an enriched property and will not be collected.

**Business Tab (Supra Databases)**

The fields on the Business Tab are identical for all Supra data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section *Business Tab (Supra Data Objects)* (page 179) for a description of the fields.

### 15.2. Defining Records

1. In the Tree View Window, right-click the Supra Database you want to define Datasets for and select *Add Supra DataSet*.

   The properties panel is displayed in the Workspace.

   ![Property Panel](image)

   This screen contains two tabs: *Technical* and *Business*.

2. Fill out the required fields.

   For a detailed description of the fields, refer to the sections:

   - *Technical Tab (Supra Datasets)* (page 162)
   - *Business Tab (Supra Data Objects)* (page 179)
3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Supra Database the new Dataset belongs to and select **Save to MetaStore**.

   **Note:** If you do not add the Dataset to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

### Technical Tab (Supra Datasets)

The following fields are available on the Technical tab:

- **Name** (page 162)
- **Size** (page 162)
- **Database** (page 162)
- **CCSID** (page 163)
- **Record Keys** (page 163)

#### Name

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

#### Database

**Note:** This field is specific for each file (sub)type. Refer to the appropriate File Access Guide for more detailed information.

This field is mandatory for IMS Segments. Enter the segment name of the record, as specified in the DBDGEN and PSBGEN statements.

For RDBMS, the Creator Name should be entered in this field.

#### Size

Mandatory field.
Enter the maximum number of characters included in the record. The value must be an integer and cannot exceed the maximum record size defined for the Dictionary File it belongs to.

**Note:** If the size is changed manually and it is too small to contain all defined fields, it will be reset by the MetaStore Manager to the smallest size needed to contain all defined fields.

**Auto Calculate**
If this option is activated, the size will be calculated automatically.

**CCSID**

Enter the Coded Character Set Identifier.
CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL.
This CCSID is an enriched property and will not be collected.

**Record Keys**

Optional field.
This option is used to identify Supra Database identification fields and the specific ranges of values for those key fields.

1. Click the **New** icon available in the Record Keys panel.

   **Note:** You can only define Record Keys after having defined Records and Fields for the Dictionary File.

   The following screen is displayed.

2. Fill out the name of the new Key.

3. Select the Key Type from the drop-down list.
The following options are available:
   - *Access*
   - *Storage*
4. Select the required Supra DataSet.
   Click the *Browse* button to display the list of all available Subschema Records and click *OK*.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the *Select Item* drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.

5. Enter the Value.

6. Apply your changes.
   The new Record Key is added to the *Record Keys* panel.

7. Repeat this action for all Record Keys you want to define.

**Business Tab (Supra Datasets)**

The fields on the Business Tab are identical for all Supra data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section *Business Tab (Supra Data Objects)* (page 179) for a description of the fields.

### 15.3. Defining Fields

1. In the Tree View Window, right-click the Supra DataSet you want to define new Fields for and select *Add Supra Field*.
   The properties panel is displayed in the Workspace.

   Two tabs are available: *Technical* and *Business*. 
2. Fill out the required fields.  
For a detailed description of the fields, refer to the sections:

- Technical Tab (Supra Fields) (page 165)
- Business Tab (Supra Data Objects) (page 179)

**Note:** Some of the values can also be modified by double-clicking or selecting them from a drop-down list in the Record Fields Window.

3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.  
In the Tree View Window, right-click the Supra Database the new Field belongs to and select **Save to MetaStore**.

**Note:** If you do not add the Field to the MetaStore Repository now, you will be asked whether you want to do this, when you leave the program.

---

**Technical Tab (Supra Fields)**

The following fields are available on the Technical tab:

- **Properties**
  - **Name** (page 166)
  - **Size** (page 166)
  - **Abs. Position** (page 167)
  - **Rel. Position** (page 167)
  - **Occurs** (page 167)
  - **Occurrence Depending On** (page 167)

- **Content**
  - **Type** (page 168)
  - **Decimals** (page 169)
  - **Unsigned** (page 170)
  - **Separated and Leading** (page 170)
  - **Initial** (page 170)
  - **Null** (page 170)
  - **Code** (page 171)
  - **Edit Mask** (page 172)
  - **Date Format** (page 174)
  - **Low Limit** (page 174)
  - **High Limit** (page 174)
  - **CCSID** (page 175)
Name

Mandatory field.
The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

Size

In the Size field, enter an integer indicating the field length as a number of bytes. If the field occurs more than once in the record, this field indicates the length of a single occurrence of the field.

<table>
<thead>
<tr>
<th>Type</th>
<th>Additional</th>
<th>Size in # Bytes</th>
<th>COBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td># characters</td>
<td>PIC A(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Character</td>
<td># characters</td>
<td>PIC X(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>National</td>
<td>2 x # characters</td>
<td>PIC N(c)</td>
<td></td>
</tr>
<tr>
<td>Varchar</td>
<td># characters</td>
<td>PIC X(c) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Binary</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) BINARY</td>
</tr>
<tr>
<td>Binary Native</td>
<td>1 &lt;= # digits &lt;= 4</td>
<td>2</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>5 &lt;= # digits &lt;= 9</td>
<td>4</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td></td>
<td>10 &lt;= # digits &lt;= 18</td>
<td>8</td>
<td>PIC S9(n)V9(d) COMP-5</td>
</tr>
<tr>
<td>Bit</td>
<td>1</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>Decimal</td>
<td>Signed / Null-signed</td>
<td>(# digits + 1) / 2</td>
<td>PIC S9(n)V9(d) PACKED-DECIMAL</td>
</tr>
<tr>
<td>Float</td>
<td>8 digits of precision</td>
<td>4</td>
<td>COMP-1</td>
</tr>
<tr>
<td></td>
<td>17 digits of precision</td>
<td>8</td>
<td>COMP-2</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>1 byte = 2 hexadecimal digits</td>
<td></td>
<td>PIC X(c) DISPLAY</td>
</tr>
<tr>
<td>Numeric</td>
<td># digits</td>
<td>PIC S9(n)V9(d) DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Printed Numeric</td>
<td># characters in Edit Mask</td>
<td>PIC EditMask DISPLAY</td>
<td></td>
</tr>
<tr>
<td>Printed Numeric</td>
<td>2 x # characters in Edit Mask</td>
<td>PIC EditMask USAGE NATIONAL</td>
<td></td>
</tr>
</tbody>
</table>
Legend:

- \( c \) = number of characters
- \( n \) = number of integer digits
- \( d \) = number of decimal digits
- \( s \) = only present when value is signed. This option is not allowed for National.

**Note:** The Record Layout window shows the default value for the Field within the selected Record. The default value matches the size of the Field, as defined in the Record Fields Window.

**Auto Calculate**

If this option is activated, the size will be calculated automatically.

**Abs. Position**

In this field, you may change the Field's starting character position that was calculated automatically by the system. The value entered must be an integer.

**Note:** If the Record's length is variable, you may not include the four-character record descriptor word when determining the start position of the Field.

**Rel. Position**

This is the 1-based position of this field within its domed structure (i.e., Record or Group).

- For a field that is not a subfield, the absolute and relative positions are equal.
- For a field \( x \) that is a subfield of group \( g \), the relative position will be calculated as \( 1 + (\text{absolute position of } x) - (\text{absolute position of } g) \).

  For example: if group \( g \) starts on position 5, and the absolute position of \( x \) is also 5, then the relative position of \( x \) is \( 1 + 5 - 5 = 1 \). In human terms: “\( x \) will start on the 1st position of \( g \)”.

**Auto Calculate**

If this option is activated, the size will be calculated automatically.

**Occurs**

Optional field.

Enter an integer indicating the maximum number of times this field can occur in the record. If you do not define a specific value, the field is assumed to occur once.

**Occurrence Depending On**

Optional field. Only applies for repeating fields (n).

Select a numeric field that is defined prior to this field within the record.

Results:
• The value you defined with the *Occurs* option represents the *maximum* number of times this field can occur.
• The value available in the selected field represents the *actual* number of times the field occurs.

1. Click the *Browse* button next to the *Occurrence Depending On* text field.
   The list of available numeric fields is displayed.

2. Select the required Field.
   The name of the selected Field is displayed in the *Occurrence Depending On* text field.

**Note:** If you click the *Browse* button again now, the Properties screen for the selected field will be displayed. If you want to select another field, you first have to delete the current field name from the text field and then click the *Browse* button.

**Type**

This field is mandatory and contains the default value *Character*.
There are two general classes of data types: *non-numeric* and *numeric*.

**Non-numeric Data Types**

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alphabetic</td>
<td>This datatype indicates a field containing strings using the characters A-Z and a-z.</td>
</tr>
<tr>
<td>Character</td>
<td>This datatype indicates that the field may contain any possible character within the character set being used (including unprintable characters). It is not possible to perform numeric operations on a Character type field, even when the field contains only digits. Matching RDBMS Data Types: CHAR, BYTE, VARCHAR, VARCHAR2, LONG, RAW, LONGVARCHAR</td>
</tr>
<tr>
<td>Graphic</td>
<td>DB2 terminology for Character datatype.</td>
</tr>
<tr>
<td>Hexadecimal</td>
<td>This datatype indicates that any hexadecimal characters are allowed within the field.</td>
</tr>
<tr>
<td>Long Varchar</td>
<td>This datatype indicates a field containing a variable character string of maximum 2 GB.</td>
</tr>
<tr>
<td>National</td>
<td>The National datatype is a subset of Unicode. The character set on which it is based is UTF-16, but restricted to two bytes per character.</td>
</tr>
<tr>
<td>Varchar</td>
<td>This datatype indicates a character field allowing the storage of values you need plus one byte for the length.</td>
</tr>
<tr>
<td>Vargraphic</td>
<td>DB2 terminology for Varchar datatype.</td>
</tr>
</tbody>
</table>

**Numeric Data Types**

<table>
<thead>
<tr>
<th>Datatype</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary</td>
<td>This datatype indicates that the field contains a binary numeric value composed of 0’s and 1’s.</td>
</tr>
</tbody>
</table>
### Decimals

If the field type allows the definition of decimals, this field contains the number of decimals.

Decimals can be entered for the following field types:
- Binary
- Binary Native
- Decimal
- Numeric
- Printed Numeric
- Printed Numeric National

If the field type does not allow the definition of decimals, this field contains the default value 0.

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Native</td>
<td>Indicates that the field contains binary native numbers. This means that its internal representation depends on the Operating System and/or processor.</td>
</tr>
<tr>
<td>Bit</td>
<td>Indicates a field occupying a single bit storage. A BIT type field may only contain the binary values 0 or 1. It is possible to perform numeric operations on a BIT type field.</td>
</tr>
<tr>
<td>Byte</td>
<td>Indicates a field occupying a single byte storage. A byte is 8 bits. It is possible to perform numeric operations on a BYTE type field.</td>
</tr>
<tr>
<td>Decimal</td>
<td>Indicates a field containing a packed decimal value. Packed decimal is the most commonly used internal numeric data type. Two decimal digits are contained in each byte of a packed decimal number. However, if the number is signed, the last half byte contains a positive or negative sign indicator.</td>
</tr>
<tr>
<td>Float</td>
<td>Indicates that the field contains floating-point numbers, encoded in an encoded exponential form.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Indicates that any the field contains decimal numbers in a printable character format, this means that a single digit is stored per byte. Leading blanks are not permitted in this field type, nor may it contain editing characters, such as commas or decimal points. Use Printed Numeric in this case. Matching RDBMS Data Types: TINYINT, SMALLINT, INTEGER, BIGINT, DECIMAL, NUMBER, FLOAT</td>
</tr>
<tr>
<td>Printed Numeric</td>
<td>Indicates that the character-based field contains a numeric value. The format in which the printed numeric value is displayed must be specified with the EDIT option.</td>
</tr>
<tr>
<td>Printed Numeric National</td>
<td>Indicates that this is a National field that contains a numeric value. The format in which the PRN-NATIONAL value is displayed must be specified with the EDIT option.</td>
</tr>
</tbody>
</table>
Unsigned

Select the *Unsigned* check box to indicate that a numeric value is not signed, i.e. that it does not contain a sign indicator (+ or -).

This field applies for the following field types:
- Binary
- Binary Native
- Numeric

**Note:** For Decimal fields, the distinction between unsigned and positive signed INPUT fields is not relevant to the system.

Separated and Leading

These two check boxes allow to define how over-punched and separate sign indicators must be treated. They only apply for the Numeric field type.

There are several possibilities.
If the sign indicator is over-punched in the number, select the *Unsigned* check box above.
- Select *Leading* to put the sign on the initial digit.
- Select *Separate* to put the sign on the last byte (no digit).
- Select *Leading* and *Separate* to put the sign in the initial byte (no digit).
If the Numeric field contains a separate plus or minus sign attached to the number, select the *Separate* check box.

Initial

Optional field.

Enter the initial value for the field.

This value is only taken into account for Target Files. If an initial value was defined for a Source file, this parameter will not affect any operation, except for syntax checking.

For Numeric fields, this is a numeric constant, where the decimal point should be given by a . (point).
For Character fields, this may be an alphanumeric constant (not enclosed by quotes) or the system fields SYS-LOW-VALUE or SYS-HIGH-VALUE, which correspond to system fields LOW-VALUE and HIGH-VALUE within COBOL.

Null

This field indicates the nullable status of the field, and whether Inbound or Outbound nulls are used.

**Note:** If this field is left blank, the default NULLABLE value defined in the MetaSuite dictionary will be applied to this field. For more information, refer to the section Create Dictionary/Enter License Key in the Generator Manager Guide.

Select the required null-indicator from the drop-down list.
**Entry** | **Description**
--- | ---
None | Select this option if no nullability information is known about the field, or when it is of no importance for the field.

**Default** | Select this option if this field is a NotNull field with a default value. The Default notion is only documentary metadata for a field. The default Nullable value defined in the MetaSuite dictionary will be applied. Refer to the section *Create a Dictionary/Enter License Key* in the Generator Manager User Guide.

**NotNull** | Select this option if a Null Value should never be allowed in this field.

**InNull** | Select this option if a Null Value is allowed in this field. The used value to define a Null value is determined by a default setting in the MetaSuite COBOL Generator. When a Null Value is assigned, the first position or character of the field itself indicates the Null Value (the so called inbound Null).

*Note:* In case of National or PRN-National, the Null Value is indicated by two bytes.

**OutNull** | Select this option if Null Values are allowed in this field. To store a Null value, an additional placeholder is foreseen in the sequential file that precedes the real field. When a Null value is assigned, this additional placeholder indicates the Null value (the so called left outbound Null). The used value to define a Null value is determined by a default setting in the MetaSuite COBOL Generator.

*Note:* This placeholder contains one byte in order to store the Null indicator, except in case of field type National or PRN-National. In that case the placeholder contains two bytes.

**OutNullR** | Select this option if Null Values are allowed in this field. To store a Null value, an additional placeholder is foreseen in the sequential file that follows the real field. When a Null value is assigned, this additional placeholder indicates the Null value (the so called right outbound Null). The used value to define a Null value is determined by a default setting in the MetaSuite COBOL Generator.

*Note:* This placeholder contains one byte in order to store the Null indicator, except in case of field type National or PRN-National. In that case the placeholder contains two bytes.

---

**Code**

Default value = *No Code*

The following options are available:

---

**Option** | **Description**
--- | ---
Code | This field is treated as a code, not as a number.

No Code | Resets the code operator to zero. Select this option when there is no additional information to be added for the field.

Time | Select this option to define a field that contains TIME information.

Timestamp | Select this option to define a field that contains TIMESTAMP information.
**Edit Mask**

This field is mandatory for Printed Numeric fields, but optional for other field types. It indicates how the alphanumeric values must be formatted.

**Note:** When the Edit Mask is set for a printed numeric field, the MetaStore Manager will reset the size of the field to the size that corresponds to the chosen Edit Mask.

Enter the characters defining a Mask in this text field. This Mask will override the default mask for the field. There is a default mask for each field type. Both the default masks and the manually created masks are composed of **Replacement** and **Insertion** characters.

The following table lists the Replacement characters and their meaning. Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field.

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>N</td>
<td>National (2-byte Unicode character set)</td>
</tr>
<tr>
<td>X</td>
<td>Character</td>
</tr>
</tbody>
</table>

The following table lists the Insertion characters and their meaning. Insertion characters indicate characters to be printed in addition to those contained in the stored field.

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
<tr>
<td>,</td>
<td>Comma (separator for the sake of large number readability or decimal separator depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>.</td>
<td>Decimal point (separator for the sake of large number readability or decimal separator, depending on the &quot;Decimal Separator&quot; option in the INI file)</td>
</tr>
<tr>
<td>B</td>
<td>Blank</td>
</tr>
<tr>
<td>-</td>
<td>Floating or trailing minus for negative values. Plus is not accepted as valid character. Positive numbers will have a blank as sign character.</td>
</tr>
<tr>
<td>+</td>
<td>Floating or trailing plus or minus sign</td>
</tr>
<tr>
<td>CR</td>
<td>Trailing credit symbol for negative values only</td>
</tr>
</tbody>
</table>
As mentioned above, there is a default Mask for each field type:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Default Mask Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields</td>
<td>The default mask contains a minus sign as the rightmost character. All negative values are printed with a trailing minus sign.</td>
</tr>
<tr>
<td>Numeric fields with decimals</td>
<td>The default mask contains a decimal point and as many digit replacement characters (9s) to the right of the decimal point as are specified by the Decimal option.</td>
</tr>
<tr>
<td>Numeric fields</td>
<td>The default mask contains as many digits as its size, without zero suppression.</td>
</tr>
<tr>
<td>Date fields</td>
<td>The default mask is its selected date format.</td>
</tr>
<tr>
<td>Alphanumeric fields</td>
<td>The default mask contains as many alphanumeric character replacement characters (X) as are required to print the field.</td>
</tr>
</tbody>
</table>

Examples of default masks:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Size</th>
<th>Default mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signed numeric fields without decimals</td>
<td>6</td>
<td>999999-</td>
</tr>
<tr>
<td>Signed numeric fields with two decimals</td>
<td>6</td>
<td>9999.99-</td>
</tr>
<tr>
<td>Character Field</td>
<td>6</td>
<td>X(6)</td>
</tr>
<tr>
<td>Unicode Field</td>
<td>6</td>
<td>N(3)</td>
</tr>
</tbody>
</table>

You may also define customized masks.

Examples:

<table>
<thead>
<tr>
<th>Field Type</th>
<th>Field Size</th>
<th>Mask</th>
<th>Field value</th>
<th>Printed value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>6</td>
<td>XXBXXX</td>
<td>AB138B</td>
<td>AB 138B</td>
</tr>
<tr>
<td>Numeric with 2 decimal positions</td>
<td>2</td>
<td>.99</td>
<td>.35</td>
<td>.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-.12</td>
<td>.12</td>
</tr>
<tr>
<td>Numeric with 3 minus signs</td>
<td>4</td>
<td>---9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>
Date Format

If the field must contain a date, select the required date format from the drop-down list. The system automatically validates date fields whenever they are referenced in a MetaMap model, and automatically converts date fields whenever they are compared to another date or used in a calculation. In all the available formats, YY or YYYY stands for the year and MM stands for the month. DD stands for the day within a month and DDD stands for the day within the year.

When the format contains a '?', this indicates the date delimiter that is used. Data formats with a '?' are only supported for Character and Varchar field types. When the data format does not contain a '?', the different parts in the date are not delimited by a special character.

The Date Format list is accessible for the following field types:

- Binary
- Binary Native
- Character
- Decimal
- Numeric
- National
- Varchar

Note: When a date format is chosen, MetaStore Manager will reset the size of the field to the size that corresponds to the chosen date format.

Low Limit

Optional field.

When a Low Limit is specified, you must define a value in the High Limit field as well. You may use this field to define a minimum value for this field. If a lower value is encountered, the system will consider the data invalid.

High Limit

Optional field.

When a High Limit is specified, you must define a value in the Low Limit field as well. You may use this field to define a maximum value for this field. If a higher value is encountered, the system will consider the data invalid.
CCSID

Enter the Coded Character Set Identifier. CCSID is used by IBM as the abbreviation for "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page. CCSID is commonly used for the data subtype CHARACTER, in order to distinguish the different character sets per country, language and the character encoding of the system. Despite of the philosophical approach of Unicode, CCSID can also be used on data type NATIONAL. This CCSID is an enriched property and will not be collected.

Business Tab (Supra Fields)

The fields on the Business Tab are identical for all Supra data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Supra Data Objects) (page 179) for a description of the fields.

15.4. Defining Relationships

1. In the Tree View Window, right-click the Supra Database you want to define Relationships for and select Add Supra Relationship.
   The properties panel is displayed in the Workspace.

Each LINK definition relates two MetaSuite (Supra Database) records to one another. The "basis" of the link is a field in one record (the "from" record, so called because data is taken from that record and used to locate the other record) and a descriptor in the other (the "to" record, so called because the descriptor leads to that record).
Two tabs are available: Technical and Business.

2. Fill out the required fields.
   For a detailed description of the fields, refer to the sections:
   - Technical Tab (Supra Relationships) (page 176)
   - Business Tab (Supra Data Objects) (page 179)
3. Save or discard your changes.

4. Save the changes to the MetaStore Repository.
   In the Tree View Window, right-click the Supra Database and select *Save to MetaStore*.

---

**Technical Tab (Supra Relationships)**

The following fields are available on the Technical tab:

- **Name** (page 176)
- **Relationship From** (page 176)
- **Relationships To** (page 178)

**Name**

Mandatory field.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters a-z, A-Z and 0-9.
- It may contain the following special characters: $, #, @, _ (underscore) and - (hyphen).
- It may not be ended with a – (hyphen).
- File names must be unique in the MetaStore Repository.

**Relationship From**

Mandatory field.
1. Fill out the name of the new Relationship key.

2. Click the Browse button next to the Relationship From text field. The following screen is displayed:

3. Select the required File Group or File and click OK.
   Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically.
   The Supra Relationship From properties panel is displayed.

4. Fill out the required fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the new Relationship</td>
</tr>
<tr>
<td>From Record</td>
<td>In order to define a particular relationship between two records, the user has to specify the first record of the relationship in the From Record field.</td>
</tr>
<tr>
<td>From Field</td>
<td>The matching field of the From Record.</td>
</tr>
</tbody>
</table>

**Note:** You can use the Browse button to display the list of available items.

5. Apply your changes.
   The name will be displayed in the Relationship From field.
Relationships To

1. Double-click the New icon available in the Relationships To text zone. The following screen is displayed:

![Relationships To Screen](image1)

2. Select the required DataSet and click OK. Two extra options are available at the top right of the pop-up window for selecting the required item:
   - Show all
     When selecting this option, the Select Item drop-down list will be deactivated and all available fields for all categories will be displayed underneath.
   - Indentation
     When selecting this option, all fields displayed will be sorted per structure instead of alphabetically. The Supra Relationship To properties panel is displayed.

![Relationship Properties Panel](image2)

3. Fill out the required fields.

   - **Name**: The name of the new Relationship
   - **To Record**: In order to define a particular relationship between two records, the user has to specify the second record of the relationship in the To Record field.
   - **To Field**: The matching field of the To Record.

   **Note**: You can use the Browse button to display the list of available items.

4. Apply your changes. The relationship key name is added in the Relationships To panel.
Business Tab (Supra Relationships)
The fields on the Business Tab are identical for all Supra data objects (Dictionary File, Record, Field, Relationship and Index). Refer to the section Business Tab (Supra Data Objects) (page 179) for a description of the fields.

15.5. Business Tab (Supra Data Objects)
The following fields are available on the Business tab:

- Business Rule (page 179)
- Note (page 179)

Note: If you want to enter text in RTF (Rich Text Format), right-click and select RTF from the context menu (or use the shortcut CTRL + R).

Business Rule
Optional field.
Enter a free-form description for the data object (Dictionary File, Record, Field, Relationship or Index). For example, the business rule or the requirements.

Note
Optional field.
Enter a free-form note for the data object.
Collecting Source and Target Dictionary Files means to capture File Definitions. This capture can be done in several ways:

- directly from the database catalog via an ODBC connection
- from a text file describing the records structures in one of the supported languages or formats.

For captures using a text file, the following formats are supported:

- COBOL Copy Book
- PL/I Include Book
- IDMS schema and IDMS record
- SAP DMI
- DB2 DDL

During the collection process, MetaStore also generates the required load and unload scripts for various RDBMSs.

**Note:** The File collection process can be customized using the INI Manager. Refer to the INI Manager User Guide for more information.
16.1. Accessing the Collect File Screen

1. On the MetaStore Toolbar, click the **Collect File** button. The following screen appears.

![Collect File Screen](image)

**Note:** You can also select the **Collect File** option on the **File** menu or use the MetaStore context menu in the Tree View Window.

The **Collect File** screen contains two tabs:

- the **Source** tab (for collecting Source Dictionary Files)
  See [Source Dictionary Files](#) on page 183.
- the **Target** tab (for collecting Target Dictionary Files)
  See [Target Dictionary Files](#) on page 212.

2. Select the **Source** tab. This tab allows you to start collecting a Source Dictionary File.

   - The options on the left match the File Types that can be used as Data Sources in MetaSuite.
   - The drop-down lists on the right match the supported formats for each supported Data Source.

The following table gives an overview of the possible selections:

<table>
<thead>
<tr>
<th>File Type</th>
<th>Supported Formats</th>
</tr>
</thead>
</table>
| Sequential| • COBOL Copy Book  
|           | • PL/I Include Book               |
| IDMS      | • Record Punch  
|           | • Schema Punch                     |
| IMS       | • COBOL Copy Book  
|           | • PL/I Include Book               |
| RDBMS     | • Catalog  
|           | • SQL DDL  
|           | • DB2 pureXML                      |
3. Select the Target tab.  
This tab allows you to start collecting a Target Dictionary File. 
- The options on the left match the File Types that can be used as Data Targets in MetaSuite. 
- The drop-down lists on the right match the supported formats for each supported Data Target. 
The following table gives an overview of the possible selections:

<table>
<thead>
<tr>
<th>File Type</th>
<th>Supported Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unload Sequential</td>
<td>• BMC Load Plus</td>
</tr>
<tr>
<td></td>
<td>• BMC UNLOAD PLUS for DB2</td>
</tr>
<tr>
<td></td>
<td>• INGRESS</td>
</tr>
<tr>
<td></td>
<td>• MS-ACCESS</td>
</tr>
<tr>
<td></td>
<td>• SESAM</td>
</tr>
<tr>
<td>XML Schema</td>
<td>XSD (standard XML Schema Description Language)</td>
</tr>
</tbody>
</table>

4. Select the required File Type and Format from the matching drop-down list.

5. Click OK. 
The rest of the procedure depends on your selection. Refer to the sections matching your selection.
A Source Dictionary File describes the metadata for the logical unit of data from which data will be extracted (data source).
The metadata describe both the physical and business characteristics of the Dictionary File. A Dictionary File contains subordinate objects, such as the underlying Records, Fields, Relationships and Indices.

17.1. File Type - Sequential

Supported File Types:
- COBOL Copy Book
- PL/I Include Book

Procedure

1. On the Collect File screen, open the Source tab and select Sequential as File Type. See Accessing the Collect File Screen on page 181.

2. Select the required format (COBOL Copy Book or PL/I Include Book) and click OK. The default folder (as defined in the MetaSuite.ini file) opens in a new window and displays the available COBOL Copy Books (extension .cbl) or PL/I Include Books (extension .pli).

Note: If the COBOL Copy Book or PL/I Include Book you need is not available in this folder, browse to the required folder.
3. Select the required COBOL Copy Book or PL/I Include Book and click Open. The following screen is displayed.

![Image of screen](image)

4. Fill out the required fields and click OK. Refer to the section Description of the Available Fields (page 184) for a detailed description of each field. The COBOL Copy Book or PL/I Include Book is collected and the matching Dictionary File Definition is displayed in the Tree View Window.

5. If required, edit the definitions. The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

6. Once the settings of the Dictionary File match your requirements, you can save it to the MetaStore. Click the Save to MetaStore icon to do so.

**Description of the Available Fields**

- [Prefix, Separator and Dictionary File Name](page 185)
- [File Type](page 185)
- [Key Field](page 185)
- [File Description Size](page 185)
- [Record Format](page 186)
- [Block Size](page 186)
- [Spanned](page 186)
- [Recording Mode](page 186)
- [Label](page 186)
- [Column Separator](page 187)
- [Row Terminator](page 187)
- [Code-control Table Name](page 187)
Prefix, Separator and Dictionary File Name

In the three fields at the top of the screen, you can enter the prefix, the separator character and the Dictionary File Name:

- The prefix will be placed in front of all Objects (file, records and fields) in order to facilitate its identification. Its maximum length is 8 characters.
- The Separator character will be placed between the defined prefix and the Dictionary File name.
- A default name for the Dictionary File is automatically filled it. You can however define another name.

**Note:** For Dictionary Files of type Unload Sequential the characters \_F are appended to the default name in order to preserve name uniqueness.

The total length of prefix, separator and filename may not exceed 32 characters.

File Type

Select the required File Type from the drop-down list. Its default value is set to Sequential. The following options are available:

- ADABAS
- Delimited
- Function
- Index
- Line Sequential
- Record Sequential
- Relative
- Sequential
- VSAM

Key Field

If required, select the Key Field for the Dictionary File from the drop-down list. Fields occurring more than once are omitted from the list. The field names are listed in alphabetical order. Their names are concatenated with the names of containing fields or records.

File Description Size

In this field, enter the maximum record size. This corresponds to the Input-Output buffer size. The value entered must be at least the size of the largest record within the file definition.
**Record Format**

Select the required format from the drop-down list.
The following options are available:

- **Fixed**: Select this option if the records of the file have all the same length. This size, expressed as a number of characters, is defined in the *File Description Size* field.

- **Undefined**: Select this option if the records in the file are no standard variable length records. Record descriptor words, which are standard for variable-length records, are not maintained at the start of each undefined record. Instead, the applications accessing the file determine the record size using other criteria. It is not possible to select this option together with the *Block Size* option. In this case, the record size defined in the following field defines the maximum number of characters contained in any record belonging to the file.

- **Variable**: Select this option if the file contains records of different sizes. The record size defined in the *File Description Size* field matches in this case the maximum record length (in number of characters).

**Block Size**

In this field, you may enter the block size. It may not be smaller than the value defined in the *File Description Size* field.

**Spanned**

Select this check box if the variable-length records may "span" two or more blocks.
This option is only applicable if the Record Format is set to *Variable*, the *Block Size* is greater than the *File Description Size*, and the subtype is index, relative or sequential.

**Recording Mode**

Select the required recording mode from the drop-down list.
The following options are available:

- Native
- ASCII
- EBCDIC

**Label**

Select the required label from the drop-down list.
The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Select <em>Standard</em> if the file has standard label records. A file label contains information such as the creation date and the length of the file.</td>
</tr>
<tr>
<td>Omit</td>
<td>Select <em>Omit</em> if the file does not contain label records.</td>
</tr>
</tbody>
</table>
Column Separator

This field contains the default Column Separator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide.

You can enter another character in this field in order to override the default Column Separator for this file.

Row Terminator

This field contains the default Row Terminator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide.

You can enter another character (sequence) in the field in order to override the default Row Terminator for this file.

Code-control Table Name

When a MetaSuite Model is generated into COBOL Source Code, this operation is performed with standard Code-control Tables.

It is possible to use a customized Code-control table instead, by entering its name in this field.

If the table name is a three-letter word, the Code-control parameter will be suffixed with INP.

Example:

Code-control XML will become XMLINP, and Code-control MQS will become MQSINP.

More information about code-control tables can be found in the Generator Manager Guide.

Database

Not relevant here. This information will be ignored.

External Source

Optional field.

You can use it in the two following situations:

- You want to define a link between the file-name definition in MetaSuite and the physical file name on disk. The physical file name will be used automatically in the generated job.

- You want to indicate that there is no direct access to the data source. The data source might be remote, or there is need to copy or transform the source file via a simple copy or via a transformation tool. In this case the file, specified by the EXTERNAL-SOURCE parameter is the file that has to be transferred or copied to the local area. The local input file will have the name indicated by file-name.

This parameter will be processed in the customizable MRL Tables. By default, MetaSuite will use the following conversion rules:
NOTE: The External Source parameter is not supported yet on every platform. UNIX and Windows are fully supported. OS390 supports the STD type. Please contact the MetaSuite support team if you want to use this option on another platform.

Remarks

In this field, enter your remarks to the Dictionary File. They will be stored as Business Rule in the MetaStore.

17.2. File Type - IDMS

Supported File Types:
- Record Punch
- Schema Punch

Procedure

1. On the Collect File screen, open the Source tab and select IDMS as File Type.
   See Accessing the Collect File Screen on page 181.

2. Select the required format (Record Punch or Schema Punch) and click OK.
   The default folder (as defined in the MetaSuite.ini file) opens in a new window and displays the available Record Punches (extension .cpy and .txt) or Schema Punches (extension .sch).

   NOTE: If the Record Punch or Schema Punch you need is not available in this folder, browse to the required folder.
3. Select the required Record Punch or Schema Punch and click Open. The following screen is displayed.

![Screenshot of the user interface](image)

4. Fill out the required fields and click OK. Refer to the section Description of the Available Fields (page 189) for a detailed description of each field. The Record Punch or Schema Punch is collected and the matching Dictionary File Definition is displayed in the Tree View Window.

5. If required, edit the definitions. The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

6. Once the settings of the Dictionary File match your requirements, you can save it to the MetaStore. Click the Save to MetaStore icon ( ) to do so.

**Description of the Available Fields**

- **Prefix, Separator and Dictionary File Name** (page 190)
- **File Type** (page 190)
- **Key Field** (page 190)
- **File Description Size** (page 190)
- **Record Format** (page 191)
- **Block Size** (page 191)
- **Spanned** (page 191)
- **Recording Mode** (page 191)
- **Label** (page 191)
- **Column Separator** (page 192)
- **Row Terminator** (page 192)
- **Code-control Table Name** (page 192)
Prefix, Separator and Dictionary File Name

In the three fields at the top of the screen, you can enter the prefix, the separator character and the Dictionary File Name:

- The prefix will be placed in front of all Objects (file, records and fields) in order to facilitate its identification. Its maximum length is 8 characters.
- The Separator character will be placed between the defined prefix and the Dictionary File name.
- A default name for the Dictionary File is automatically filled it. You can however define another name.

**Note:** For Dictionary Files of type Unload Sequential the characters _F are appended to the default name in order to preserve name uniqueness.

The total length of prefix, separator and filename may not exceed 32 characters.

File Type

Select the required File Type from the drop-down list. Its default value is set to Sequential. The following options are available:

- ADABAS
- Delimited
- Function
- Index
- Line Sequential
- Record Sequential
- Relative
- Sequential
- VSAM

Key Field

If required, select the Key Field for the Dictionary File from the drop-down list. Fields occurring more than once are omitted from the list. The field names are listed in alphabetical order. Their names are concatenated with the names of containing fields or records.

File Description Size

In this field, enter the maximum record size. This corresponds to the Input-Output buffer size. The value entered must be at least the size of the largest record within the file definition.
Record Format

Select the required format from the drop-down list.
The following options are available:

- **Fixed**: Select this option if the records of the file have all the same length. This size, expressed as a number of characters, is defined in the *File Description Size* field.
- **Undefined**: Select this option if the records in the file are no standard variable length records. Record descriptor words, which are standard for variable-length records, are not maintained at the start of each undefined record. Instead, the applications accessing the file determine the record size using other criteria. It is not possible to select this option together with the *Block Size* option. In this case, the record size defined in the following field defines the maximum number of characters contained in any record belonging to the file.
- **Variable**: Select this option if the file contains records of different sizes. The record size defined in the *File Description Size* field matches in this case the maximum record length (in number of characters).

Block Size

In this field, you may enter the block size. It may not be smaller than the value defined in the *File Description Size* field.

Spanned

Select this check box if the variable-length records may "span" two or more blocks.
This option is only applicable if the Record Format is set to *Variable*, the *Block Size* is greater than the *File Description Size*, and the subtype is index, relative or sequential.

Recording Mode

Select the required recording mode from the drop-down list.
The following options are available:

- **Native**
- **ASCII**
- **EBCDIC**

Label

Select the required label from the drop-down list.
The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Select <em>Standard</em> if the file has standard label records. A file label contains information such as the creation date and the length of the file.</td>
</tr>
<tr>
<td>Omit</td>
<td>Select <em>Omit</em> if the file does not contain label records.</td>
</tr>
</tbody>
</table>
Column Separator

This field contains the default Column Separator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide. You can enter another character in this field in order to override the default Column Separator for this file.

Row Terminator

This field contains the default Row Terminator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide. You can enter another character (sequence) in the field in order to override the default Row Terminator for this file.

Code-control Table Name

When a MetaSuite Model is generated into COBOL Source Code, this operation is performed with standard Code-control Tables. It is possible to use a customized Code-control table instead, by entering its name in this field. If the table name is a three-letter word, the Code-control parameter will be suffixed with INP. Example: Code-control XML will become XMLINP, and Code-control MQS will become MQSINP. More information about code-control tables can be found in the Generator Manager Guide.

Database

Enter the name of the corresponding database.

External Source

Optional field. You can use it in the two following situations:

- You want to define a link between the file-name definition in MetaSuite and the physical file name on disk. The physical file name will be used automatically in the generated job.
- You want to indicate that there is no direct access to the data source. The data source might be remote, or there is need to copy or transform the source file via a simple copy or via a transformation tool. In this case the file, specified by the EXTERNAL-SOURCE parameter is the file that has to be transferred or copied to the local area. The local input file will have the name indicated by file-name.

This parameter will be processed in the customizable MRL Tables. By default, MetaSuite will use the following conversion rules:
### External Source Name | Meaning
--- | ---
STD:Filename | This file name will be taken as input file. (This is the first method of using EXTERNAL-SOURCE)
FTP:Filename | The file will be downloaded via the FTP protocol.
CPY:Filename | A file copy will be done from this file to the standard file.

**Note:** The External Source parameter is not supported yet on every platform. UNIX and Windows are fully supported. OS390 supports the STD type. Please contact the MetaSuite support team if you want to use this option on another platform.

### Remarks

In this field, enter your remarks to the Dictionary File. They will be stored as Business Rule in the MetaStore.

### 17.3. File Type - IMS

Supported File Types:
- COBOL Copy Book
- PL/I Include Books

**Procedure**

1. On the Collect File screen, open the Source tab and select IMS as File Type. See [Accessing the Collect File Screen](#) on page 181.

2. Select the required format (COBOL Copy Book or PL/I Include Book) and click **OK**.
   The default folder (as defined in the MetaSuite.ini file) opens in a new window and displays the available COBOL Copy Books or PL/I Include Books.

   **Note:** If the COBOL Copy Book or PL/I Include Book you need is not available in this folder, browse to the required folder.

3. Select the required COBOL Copy Book or PL/I Include Book and click **Open**.
   The following screen is displayed.
4. Fill out the required fields and click OK.
   Refer to the section Description of the Available Fields (page 194) for a detailed description of each field. The COBOL Copy Book or PL/I Include Book is collected and the matching Dictionary File Definition is displayed in the Tree View Window.

5. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

6. Once the settings of the Dictionary File match your requirements, you can save it to the MetaStore.
   Click the Save to MetaStore icon to do so.

**Description of the Available Fields**

- **File Name** (page 194)
- **PCB Name** (page 194)

**File Name**

File name for the ADD FILE command, reference name for the program.

**PCB Name**

Program view of the database (program communication block).

**17.4. File Type - RDBMS**

Supported File Types:

- Catalog
- SQL DDL
- DB2 pureXML (not supported yet)

**Procedures**

- **Catalog** (page 194)
- **SQL DDL** (page 197)

**Catalog**

1. On the Collect File screen, open the Source tab and select RDBMS as File Type. See Accessing the Collect File Screen on page 181.
2. Select Catalog as supported format and click OK. The RDBMS Logon screen is displayed.

3. Fill out the required fields and click OK.

   Field Description
   
   **User ID** Enter your User Name to get access to the database. The User ID displayed in this field is the default User ID defined in Metasuite.ini.
   
   **Password** Enter your password.
   
   **Data Source Name** Enter the required DSN. The DSN displayed in this field is the default DSN defined in Metasuite.ini.
   
   **Database** Enter the name of the database.

   **Note:** If required, make the necessary selections to obtain the ODBC connection.

   Once the ODBC connection is established, the following screen is displayed.

4. In the upper part of the screen, define your selection criteria and click Show. The lower part of the screen will list the Tables and/or Views matching your selection criteria.
Note: Owner and Qualifier are ODBC terms. The meaning of these terms can differ between different databases or different ODBC drivers. Some drivers return an error when you enter a search string for Owner or Qualifier or some return no tables. If you encounter problems when entering search criteria for Owner or Qualifier, try entering only an asterisk.

If the User ID you used to access the database does not have the required access rights to some or all Tables or Views, these Tables and Views will not be displayed.

5. Select the Table(s) and View(s) you want to include and click OK.

Note: If you want to select multiple adjacent items, click the first and the last item while pressing the Shift Key.

If you want to select multiple non-adjacent items, click the items while pressing the Ctrl Key.

The following screen is displayed.

6. Fill out the required fields and click OK.

The following fields are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter the Dictionary File Name for the ADD FILE statement in MDL-file.</td>
</tr>
<tr>
<td>Subtype</td>
<td>Select the required SubType from the drop-down list. You can also select Generator Default to use the SQL dialect set as default in the Generator Manager.</td>
</tr>
</tbody>
</table>
The Dictionary File Definition is displayed in the Tree View Window.

7. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

8. Once the settings of the Dictionary File match your requirements, you can save it to the MetaStore.
   Click the Save to MetaStore button ( ) to do so.

**SQL DDL**

1. On the Collect File screen, open the Source tab and select RDBMS as File Type.
   See Accessing the Collect File Screen on page 181.

2. Select SQL DDL as supported format and click OK.
   The default folder (as defined in the MetaSuite.ini file) opens in a new window and displays the available SQL DDLs (extension .ddl).

   **Note:** If the DDL you need is not available in this folder, browse to the required folder.

3. Select the required DDL and click Open.
   The following screen is displayed.

4. Fill out the required fields and click OK.
   The following fields are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uncommitted Read</td>
<td>(DB2 only) Select this check box, if you want to perform a so-called dirty read. This means reading data without checking if the data has been locked or not.</td>
</tr>
<tr>
<td>Fetch First Row Only</td>
<td>(DB2 only) Select this check box, if you want to read only the first row, if different rows contain the same key.</td>
</tr>
</tbody>
</table>
The screen listing the available Tables and Views is displayed.

5. Select the Table(s) and View(s) you want to include in the new Dictionary File and click OK.

The screen lists the Tables and Views belonging to the selected Database. If you want to select multiple adjacent items, click the first and the last item while pressing the Shift Key. If you want to select multiple non-adjacent items, click the items while pressing the Ctrl Key. The Dictionary File for each selected Table or Views is displayed in the Tree View Window.

6. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

7. Once the settings of the Dictionary File match your requirements, you can save it to the MetaStore.
   Click the Save to MetaStore icon ( ) to do so.

17.5. File Type - Unload Sequential
Supported File Types:
• BMC Load Plus
• BMS UNLOAD PLUS for DB2
• INGRES
• MS-Access
• SESAM

Procedures
• BMC and INGRES (page 199)
• MS-Access and SESAM (page 202)

### BMC and INGRES

<table>
<thead>
<tr>
<th>BMC</th>
<th>BMC is a tool facilitating the unloading of DB2 files. Use the option Unload Sequential on the Source tab of the Collect File screen to collect a BMC Dictionary File. During the collection process, MetaStore generates the required unload script matching the metadata found in the source collected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>INGRES</td>
<td>If you prefer retrieving information from an unloaded sequential file, rather than directly from an CA/Ingres Relational Database, you can use the option Unload Sequential. MetaSuite will generate the Dictionary File matching the metadata of the unloaded sequential file. Furthermore, the required unload script will be generated, if required.</td>
</tr>
</tbody>
</table>

1. **On the Collect File screen, open the Source tab and select Unload Sequential as File Type.**
   See [Accessing the Collect File Screen](#) on page 181.
   If required, make the necessary selections to obtain the ODBC connection.

2. **Select BMC or CA/Ingres as supported format and click OK.**
   The Unload Sequential Logon screen is displayed.

**Note:** If an error is displayed, please check the settings in MetaSuite.ini. Refer to the INI Manager Guide for more information.
3. Fill out the required fields and click OK.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Enter your User Name to get access to the database. The User ID displayed in this field is the default User ID defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter your password.</td>
</tr>
<tr>
<td>Data Source Name</td>
<td>Enter the required DSN. The DSN displayed in this field is the default DSN defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the database.</td>
</tr>
</tbody>
</table>

**Note:** If required, make the necessary selections to obtain the ODBC connection.

Once the ODBC connection is established, the following screen is displayed.

4. In the upper part of the screen, define your selection criteria and click Show.
   The lower part of the screen will list the Tables and/or Views matching your selection criteria.
5. Select the Table(s) and View(s) you want to include and click OK.

**Note:** If you want to select multiple adjacent items, click the first and the last item while pressing the Shift Key.
If you want to select multiple non-adjacent items, click the items while pressing the Ctrl Key.

The following screen is displayed for each selected Table or View.

Each selected Table or View will be saved in a separate Dictionary File.
6. Fill out the required fields and click OK.
   Refer to Description of the Available Fields (page 205) for a detailed description of each field.

7. Confirm the default location for the SQL Unload Script or browse to another folder and click OK.

8. Repeat the previous steps for each selected Table or View.
   The Dictionary File for each selected Table or Views is displayed in the Tree View Window.

9. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

10. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.
    Click the Save to MetaStore icon ( ) to do so.

11. In order to really perform the Unload, you need to further process the MXL file.
    See Processing the MXL File on page 209.

**MS-Access and SESAM**

If you prefer retrieving information from an unloaded sequential file, rather than directly from an RDBMS, you can use the option Unload Sequential. MetaSuite will generate the Dictionary File matching the metadata of the unloaded sequential file.

1. On the Collect File screen, open the Source tab and select Unload Sequential as File Type.
   See Accessing the Collect File Screen on page 181.

2. Select MS-Access or Sesam as supported format and click OK.
   The Unload Sequential Logon screen is displayed.

![MS-ACCESS Unload Sequential Logon](image)

**Note:** If an error is displayed, please check the settings in MetaSuite.ini. Refer to the INI Manager Guide for more information.
3. Fill out the required fields and click OK.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Enter your User Name to get access to the database. The User ID displayed in this field is the default User ID defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter your password.</td>
</tr>
<tr>
<td>Data Source Name</td>
<td>Enter the required DSN. The DSN displayed in this field is the default DSN defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the database.</td>
</tr>
</tbody>
</table>

**Note:** If required, make the necessary selections to obtain the ODBC connection.

Once the ODBC connection is established, the following screen is displayed.

4. In the upper part of the screen, define your selection criteria and click Show. The lower part of the screen will list the Tables and/or Views matching your selection criteria.
Note: Owner and Qualifier are ODBC terms. The meaning of these terms can differ between different databases or different ODBC drivers. Some drivers return an error when you enter a search string for Owner or Qualifier or some return no tables.

If you encounter problems when entering search criteria for Owner or Qualifier, try entering only an asterisk.

If the User ID you used to access the database does not have the required access rights to some or all Tables or Views, these Tables and Views will not be displayed.

5. Select the Table(s) and View(s) you want to include and click OK.

Note: If you want to select multiple adjacent items, click the first and the last item while pressing the Shift Key.

If you want to select multiple non-adjacent items, click the items while pressing the Ctrl Key.

The following screen is displayed for each selected Table or View.

Each selected Table or View will be saved in a separate Dictionary File.
6. Fill out the required fields and click OK.
   Refer to Description of the Available Fields (page 205) for a detailed description of each field.

7. Repeat the previous steps for each selected Table or View.
   The Dictionary File for each selected Table or Views is displayed in the Tree View Window.

8. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

9. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.
   Click the Save to MetaStore icon ( ) to do so.

10. In order to really perform the Unload, you need to further process the MXL file.
    See Processing the MXL File on page 209.

Description of the Available Fields

- Prefix, Separator and Dictionary File Name (page 205)
- File Type (page 206)
- Key Field (page 206)
- File Description Size (page 206)
- Record Format (page 206)
- Block Size (page 206)
- Spanned (page 207)
- Recording Mode (page 207)
- Label (page 207)
- Column Separator (page 207)
- Row Terminator (page 207)
- Code-control Table Name (page 207)
- Database (page 208)
- External Source (page 208)
- Remarks (page 208)

Prefix, Separator and Dictionary File Name

In the three fields at the top of the screen, you can enter the prefix, the separator character and the Dictionary File Name:

- The prefix will be placed in front of all Objects (file, records and fields) in order to facilitate its identification. Its maximum length is 8 characters.
- The Separator character will be placed between the defined prefix and the Dictionary File name.
A default name for the Dictionary File is automatically filled in. You can however define another name.

**Note:** For Dictionary Files of type *Unload Sequential* the characters \_F are appended to the default name in order to preserve name uniqueness.

The total length of prefix, separator and filename may not exceed 32 characters.

**File Type**

This read-only Field is always set to *Sequential*.

**Key Field**

If required, select the Key Field for the Dictionary File from the drop-down list. Fields occurring more than once are omitted from the list. The field names are listed in alphabetical order. Their names are concatenated with the names of containing fields or records.

**File Description Size**

In this field, enter the maximum record size. This corresponds to the Input-Output buffer size.

The value entered must be at least the size of the largest record within the file definition.

**Record Format**

Select the required format from the drop-down list.

The following options are available:

- **Fixed**: Select this option if the records of the file have all the same length. This size, expressed as a number of characters, is defined in the *File Description Size* field.

- **Undefined**: Select this option if the records in the file are no standard variable length records. Record descriptor words, which are standard for variable-length records, are not maintained at the start of each undefined record. Instead, the applications accessing the file determine the record size using other criteria. It is not possible to select this option together with the *Block Size* option. In this case, the record size defined in the following field defines the maximum number of characters contained in any record belonging to the file.

- **Variable**: Select this option if the file contains records of different sizes. The record size defined in the *File Description Size* field matches in this case the maximum record length (in number of characters)

**Block Size**

In this field, you may enter the block size. It may not be smaller than the value defined in the *File Description Size* field.
Spanned

Select this check box if the variable-length records may "span" two or more blocks. This option is only applicable if the Record Format is set to Variable, the Block Size is greater than the File Description Size, and the subtype is index, relative or sequential.

Recording Mode

Select the required recording mode from the drop-down list. The following options are available:

- Native
- ASCII
- EBCDIC

Label

Select the required label from the drop-down list. The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Select Standard if the file has standard label records. A file label contains information such as the creation date and the length of the file.</td>
</tr>
<tr>
<td>Omit</td>
<td>Select Omit if the file does not contain label records.</td>
</tr>
</tbody>
</table>

Column Separator

This field contains the default Column Separator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide. You can enter another character in this field in order to override the default Column Separator for this file.

Row Terminator

This field contains the default Row Terminator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide. You can enter another character (sequence) in the field in order to override the default Row Terminator for this file.

Code-control Table Name

When a MetaSuite Model is generated into COBOL Source Code, this operation is performed with standard Code-control Tables.
It is possible to use a customized Code-control table instead, by entering its name in this field. If the table name is a three-letter word, the Code-control parameter will be suffixed with INP. Example: Code-control XML will become XMLINP, and Code-control MQS will become MQSINP. More information about code-control tables can be found in the *Generator Manager Guide*.

**Database**

Informational. The name of the Database that will be unloaded.

**External Source**

Optional field.

You can use it in the two following situations:

- You want to define a link between the *file-name* definition in MetaSuite and the physical file name on disk. The physical file name will be used automatically in the generated job.
- You want to indicate that there is no direct access to the data source. The data source might be remote, or there is need to copy or transform the source file via a simple copy or via a transformation tool. In this case the file, specified by the *EXTERNAL-SOURCE* parameter is the file that has to be transferred or copied to the local area. The local input file will have the name indicated by *file-name*.

This parameter will be processed in the customizable MRL Tables. By default, MetaSuite will use the following conversion rules:

<table>
<thead>
<tr>
<th>External Source Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STD:Filename</td>
<td>This file name will be taken as input file. (This is the first method of using EXTERNAL-SOURCE)</td>
</tr>
<tr>
<td>FTP:Filename</td>
<td>The file will be downloaded via the FTP protocol.</td>
</tr>
<tr>
<td>CPY:Filename</td>
<td>A file copy will be done from this file to the standard file.</td>
</tr>
</tbody>
</table>

**Note:** The External Source parameter is not supported yet on every platform. UNIX and Windows are fully supported. OS390 supports the STD type. Please contact the MetaSuite support team if you want to use this option on another platform.

**Remarks**

In this field, enter your remarks to the Dictionary File. They will be stored as Business Rule in the MetaStore.
Processing the MXL File

This section describes the steps to be performed once the Dictionary Files and MXL program files have been collected, in order to effectively unload Sequential files from a database.

1. Start the Generator Manager:
   Select the Generator type and click OK to display the Generator Manager Window.

2. On the Generate tab, select the MXL type and next the required MXL file, and click the Generate icon.
   For more information, refer to the Generator Manager User Guide.
   Results:
   - The COBOL Source Code is generated and saved (by default) in the $<MetaSuiteRoot>/MGL folder. It has the same name as the MXL file, but the extension is .MGL.
   - The MetaSuite Run Script is generated and saved (by default) in the $<MetaSuiteRoot>/MRL folder. It has the same name as the MXL file, but the extension is .MRL.

**Note:** Use the Other Functions tab to access the folders identified in the Browse section.
3. Transfer both the MGL and MRL files to the platform where the program needs to be executed.

4. You may need to edit the MRL file to change the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPTTD01</td>
<td>In this field you will find the name of the Data Target File. By default, this file has the same name as the MRL file, but the extension is .d01. You may change the name and the extension of the file. You may also enter a path in front of the file name, if you want to save the Data Target File in another directory. In order to change the target file name, you can also set the MTL option OPTION-USE-TARGETNAME to ON.</td>
</tr>
<tr>
<td>PPTDBG</td>
<td>In this field you will find the name of the Debug File. All debug information generated during the program execution will be saved in this file. By default, this file has the same name as the MRL file, but the extension is .dbg. You may change the name and the extension of the file. You may also enter a path in front of the file name, if you want to save the Data Target File in another directory.</td>
</tr>
<tr>
<td>PPTLOG</td>
<td>In this field you will find the name of the Log File. This file contains information logged during the program execution. However, it is not saved in a readable format. By default, this file has the same name as the MRL file, but the extension is .log. You may change the name and the extension of the file. You may also enter a path in front of the file name, if you want to save the Data Target File in another directory.</td>
</tr>
<tr>
<td>PPTLST</td>
<td>In this field you will find the name of the Listing File. This file contains information logged during the program execution and it is in readable format. By default, this file has the same name as the MRL file, but the extension is .lst. You may change the name and the extension of the file. You may also enter a path in front of the file name, if you want to save the Data Target File in another directory.</td>
</tr>
</tbody>
</table>

**Note:** You can also change the MXL tables in order to perform the changes automatically.

5. Compile and link the MGL file.
   - To compile this file, you can for example use the compile script in the `<MetaSuiteRoot>/TMP` folder which is defined on the MetaMap Manager Settings screen (i.e., mkcob.cmd).
   
   Copy the script to the MGL folder and adapt the access to the corresponding compiler on the Target platform. Next, execute the following command:
   
   ```
   mkcob.cmd file_name (do not add the extension .MGL)
   ```
   
   - To create the link to the database, adapt the .MRL file before starting the processing.

6. You are now ready to execute the Run Script. The Target Data File will be generated.
17.6. File Type: XML Schema

XSD is a file format, describing a structure. This format is used to validate an XML file against a specific template.

The MetaStore collect function extracts the file/record/field information from an XSD file. The resulting dictionary file has the XML subtype, meaning that the source file is in XML format.


2. On the Source tab, select XML Schema as File Type and click OK.

3. Select the file(s) you want to collect and click OK.

4. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

5. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.
   Click the Save to MetaStore icon ( ) to do so.
A Target Dictionary File describes the metadata for the logical unit of data to which data will be written (data target).

The metadata describe both the physical and business characteristics of the Dictionary File. A Dictionary File contains subordinate objects, such as the underlying Records, Fields, Relationships and Indices.

When collecting target dictionary files, the data definition is created, as well as an upload script to load the data definition in the target database.

18.1. File Type: Load Sequential

Supported File Types:

- BMC Load Plus
- BMC UNLOAD PLUS for DB2
- CA Fast Unload for DB2 for z/OS
- DB2 for OS/400
- DB2 for z/OS
- DB2 LUW
- Teradata
- SAP/R3-DMI 3.1
- SAP/R3-DMI 4.0

Procedures

During the collection process, MetaStore generates the required load script matching the data found in the source collected.

Refer to the following sections for the description of the collection process:

- Procedure (BMC, DB2/xxx) (page 213)
- Procedure (SAP) (page 215)
- Procedure (Teradata) (page 217)
Procedure (BMC, DB2/xxx)

1. On the Collect File screen, open the Target tab and select Load Sequential as File Type.

2. Select the required format from the drop-down list and click OK. The Load Sequential Logon screen is displayed.

3. Fill out the required fields and click OK.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Enter your User Name to get access to the database. The User ID displayed in this field is the default User ID defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter your password.</td>
</tr>
<tr>
<td>Data Source Name</td>
<td>Enter the required DSN. The DSN displayed in this field is the default DSN defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the database.</td>
</tr>
</tbody>
</table>

**Note:** If required, make the necessary selections to obtain the ODBC connection.

Once the ODBC connection is established, the following screen is displayed.
4. In the upper part of the screen, define your selection criteria and click Show.
The lower part of the screen will list the Tables and/or Views matching your selection criteria.

<table>
<thead>
<tr>
<th>Selection Criterion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a Table or View name in this field, if you want to limit your search to tables or views with this name. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>Enter a Qualifier in this field, if you want to limit your search to tables or views with this Qualifier. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Owner</td>
<td>Enter a User ID in this field, if you want to limit your search to tables or views owned by this User ID. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
</tbody>
</table>
| Type                | Select one of the following types:  
  - TABLE: include database tables in your search.  
  - VIEW: include database views in your search.  
  - BOTH: include both database tables and views in your search. |

**Note:** Owner and Qualifier are ODBC terms. The meaning of these terms can differ between different databases or different ODBC drivers. Some drivers return an error when you enter a search string for Owner or Qualifier or some return no tables. If you encounter problems when entering selection criteria for Owner or Qualifier, try entering only an asterisk. If the User ID you used to access the database does not have the required access rights to some or all Tables or Views, these Tables and Views will not be displayed.

5. Select the Table(s) and View(s) you want to include in the new Dictionary Files and click OK.

**Note:** If you want to select multiple adjacent items, click the first and the last item while pressing the Shift Key. If you want to select multiple non-adjacent items, click the items while pressing the Ctrl Key.

The following screen is displayed for each selected Table or View.

Each selected Table or View will be saved in a separate Dictionary File.
6. Fill out the required fields and click OK.
Refer to the section Description of the Available Fields (page 219) for a detailed description of each field.
The Load Sequential Options screen will be displayed.

![Load Sequential Options](image)

7. Define the Load Options for the selected table and click OK.
Refer to your database-specific technical documentation for more information about the displayed options.

**Note:** This step is not needed for DB2/400.

8. Specify the name and location for the Load Script and click Save.

9. Repeat the previous steps for each selected Table or View.
The Dictionary File for each selected Table or Views is displayed in the Tree View Window.

10. If required, edit the definitions.
The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

11. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.
Click the **Save to MetaStore** icon ( ) to do so.

**Procedure (SAP)**

1. On the Collect File screen, open the Target tab and select Load Sequential as File Type.

2. Select the required format from the drop-down list and click OK.
The default folder (as defined in the MetaSuite.ini file) opens in a new window and displays the available DMI files (extensions .idoc).

**Note:** If the DMI file you need is not available in this folder, browse to the required folder.
3. Select the required DMI file and click OK.
   The following screen is displayed.

4. Fill out the required fields and click OK.
   Refer to the section Description of the Available Fields (page 219) for a detailed description of each field.
   The DMI file is collected and the matching Dictionary File Definition is displayed in the Tree View Window.

5. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

6. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.
   Click the Save to MetaStore icon ( ) to do so.
Procedure (Teradata)

1. On the Collect File screen, open the Target tab and select Load Sequential as File Type.

2. Select the required format from the drop-down list and click OK.
   The Load Sequential Logon screen is displayed.

3. Fill out the required fields and click OK.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Enter your User Name to get access to the database. The User ID displayed in this field is the default User ID defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter your password.</td>
</tr>
<tr>
<td>Data Source Name</td>
<td>Enter the required DSN. The DSN displayed in this field is the default DSN defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the database.</td>
</tr>
</tbody>
</table>

**Note:** If required, make the necessary selections to obtain the ODBC connection.

Once the ODBC connection is established, the following screen is displayed.
4. In the upper part of the screen, define your selection criteria and click Show. The lower part of the screen will list the Tables and/or Views matching your selection criteria.

<table>
<thead>
<tr>
<th>Selection Criterion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a Table or View name in this field, if you want to limit your search to tables or views with this name. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>Enter a Qualifier in this field, if you want to limit your search to tables or views with this Qualifier. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Owner</td>
<td>Enter a User ID in this field, if you want to limit your search to tables or views owned by this User ID. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
</tbody>
</table>
| Type                | Select one of the following types:  
  • TABLE: include database tables in your search.  
  • VIEW: include database views in your search.  
  • BOTH: include both database tables and views in your search. |

**Note:** Owner and Qualifier are ODBC terms. The meaning of these terms can differ between different databases or different ODBC drivers. Some drivers return an error when you enter a search string for Owner or Qualifier or some return no tables. If you encounter problems when entering selection criteria for Owner or Qualifier, try entering only an asterisk. If the User ID you used to access the database does not have the required access rights to some or all Tables or Views, these Tables and Views will not be displayed.

5. Select the Table(s) and View(s) you want to include in the new Dictionary Files and click OK.

**Note:** If you want to select multiple adjacent items, click the first and the last item while pressing the Shift Key.

If you want to select multiple non-adjacent items, click the items while pressing the Ctrl Key.

The following screen is displayed for each selected Table or View.

Each selected Table or View will be saved in a separate Dictionary File.
6. Fill out the required fields and click OK.
Refer to the section Description of the Available Fields (page 219) for a detailed description of each field.
The following screen will be displayed.

![Teradata Load Sequential Options dialog box]

7. Define the Teradata Options and click OK.
The following fields are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restart Log Table</td>
<td>Enter the name of the Log Table.</td>
</tr>
<tr>
<td>User Name</td>
<td>Enter the Username</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the Password</td>
</tr>
<tr>
<td>Checkpoint Interval</td>
<td>Enter the checkpoint interval related to a possible restart.</td>
</tr>
<tr>
<td>Infile</td>
<td>Enter the Input File name.</td>
</tr>
</tbody>
</table>

8. Confirm the location for the Load Script or browse to the required location first. Then click OK.

9. Repeat the previous steps for each selected Table or View.
The Dictionary File for each selected Table or View is displayed in the Tree View Window.

10. If required, edit the definitions.
The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

11. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.
Click the Save to MetaStore icon ( ) to do so.

Description of the Available Fields

- **Prefix, Separator and Dictionary File Name** (page 220)
- **File Type** (page 220)
- **Key Field** (page 220)
- **File Description Size** (page 220)
- **Record Format** (page 220)
- **Block Size** (page 221)
Prefix, Separator and Dictionary File Name

In the three fields at the top of the screen, you can enter the prefix, the separator character and the Dictionary File Name:

- The prefix will be placed in front of all Objects (file, records and fields) in order to facilitate its identification. Its maximum length is 8 characters.
- The Separator character will be placed between the defined prefix and the Dictionary File name.
- A default name for the Dictionary File is automatically filled in. The characters _F are appended to this name in order to preserve name uniqueness. You can however define another name.

The total length of prefix, separator and filename may not exceed 32 characters.

File Type

This read-only Field is always set to Sequential.

Key Field

If required, select the Key Field for the Dictionary File from the drop-down list. Fields occurring more than once are omitted from the list. The field names are listed in alphabetical order. Their names are concatenated with the names of containing fields or records.

File Description Size

In this field, enter the maximum record size. This corresponds to the Input-Output buffer size. The value entered must be at least the size of the largest record within the file definition.

Record Format

Select the required format from the drop-down list. The following options are available:

- **Fixed**: Select this option if the records of the file have all the same length. This size, expressed as a number of characters, is defined in the File Description Size field.
• **Undefined**: Select this option if the records in the file are no standard variable length records. Record descriptor words, which are standard for variable-length records, are not maintained at the start of each undefined record. Instead, the applications accessing the file determine the record size using other criteria. It is not possible to select this option together with the **Block Size** option. In this case, the record size defined in the following field defines the maximum number of characters contained in any record belonging to the file.

• **Variable**: Select this option if the file contains records of different sizes. The record size defined in the **File Description Size** field matches in this case the maximum record length (in number of characters).

### Block Size

In this field, you may enter the block size. It may not be smaller than the value defined in the **File Description Size** field.

### Spanned

Select this check box if the variable-length records may "span" two or more blocks.

This option is only applicable if the Record Format is set to **Variable**, the **Block Size** is greater than the **File Description Size**, and the subtype is index, relative or sequential.

### Recording Mode

Select the required recording mode from the drop-down list.

The following options are available:

• **Native**
• **ASCII**
• **EBCDIC**

### Label

Select the required label from the drop-down list.

The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>Select Standard if the file has standard label records. A file label contains information such as the creation date and the length of the file.</td>
</tr>
<tr>
<td>Omit</td>
<td>Select Omit if the file does not contain label records.</td>
</tr>
</tbody>
</table>

### Column separator

This field contains the default Column Separator for Standard Files, if defined on the **MetaMap Manager Settings** tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide.
You can enter another character in this field in order to override the default Column Separator for this file.

**Row terminator**

This field contains the default Row Terminator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter *Customizing the MetaSuite INI Settings* in the *Install and Setup Guide*. You can enter another character (sequence) in the field in order to override the default Row Terminator for this file.

**Code-control Table Name**

When a MetaSuite Model is generated into COBOL Source Code, this operation is performed with standard Code-control Tables.

It is possible to use a customized code-control table instead, by entering its name in this field.

If the table name is a three-letter word, the Code-Control parameter will be suffixed with OUT.

Example:

Code-control XML will become XMLOUT, and Code-control MQS will become MQSOUT

More information about code-control tables can be found in the *Generator Manager Guide*.

**Database**

Name of the database you want to feed.

**External Source**

Optional field.

You can use it in the two following situations:

- You want to define a link between the *file-name* definition in MetaSuite and the physical file name on disk. The physical file name will be used automatically in the generated job.

- You want to indicate that there is no direct access to the data source. The data source might be remote, or there is need to copy or transform the source file via a simple copy or via a transformation tool. In this case the file, specified by the *EXTERNAL-SOURCE* parameter is the file that has to be transferred or copied to the local area. The local input file will have the name indicated by *file-name*.

This parameter will be processed in the customizable MRL Tables. By default, MetaSuite will use the following conversion rules:
External Source Name | Meaning
--- | ---
STD:Filename | This file name will be taken as input file. (This is the first method of using EXTERNAL-SOURCE)
FTP:Filename | The file will be downloaded via the FTP protocol.
CPY:Filename | A file copy will be done from this file to the standard file.

**Note:** The External Source parameter is not supported yet on every platform. UNIX and Windows are fully supported. OS390 supports the STD type. Please contact the MetaSuite support team if you want to use this option on another platform.

**Remarks**

In this field, enter your remarks to the Dictionary File. They will be stored as Business Rule in the MetaStore.

### 18.2. File Type: Load Delimited

Supported File Types:
- ADABAS D
- DB2/2 LUW
- Informix
- Oracle
- Red Brick
- SQL Server
- Sybase

The option *Load Delimited* on the *Target* tab of the *Collect File* screen allows you to generate a Copy script to load the stored data from a delimited file. All tables are found in the existing data sources.
Procedure

1. On the Collect File screen, open the Target tab and select Load Delimited as File Type.

2. Select the required format from the drop-down list and click OK. The Load Delimited Logon screen is displayed.

3. Fill out the required fields and click OK.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Enter your User Name to get access to the database. The User ID displayed in this field is the default User ID defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter your password.</td>
</tr>
<tr>
<td>Data Source Name</td>
<td>Enter the required DSN. The DSN displayed in this field is the default DSN defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the database.</td>
</tr>
</tbody>
</table>

**Note:** If required, make the necessary selections to obtain the ODBC connection.

Once the ODBC connection is established, the following screen is displayed.
4. In the upper part of the screen, define your selection criteria and click Show. The lower part of the screen will list the Tables and/or Views matching your selection criteria.

<table>
<thead>
<tr>
<th>Selection Criteria</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a Table or View name in this field, if you want to limit your search to tables or views with this name. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>Enter a Qualifier in this field, if you want to limit your search to tables or views with this Qualifier. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Owner</td>
<td>Enter a User ID in this field, if you want to limit your search to tables or views owned by this User ID. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
</tbody>
</table>
| Type               | Select one of the following types:  
  • TABLE: include database tables in your search.  
  • VIEW: include database views in your search.  
  • BOTH: include both database tables and views in your search. |

**Note:** Owner and Qualifier are ODBC terms. The meaning of these terms can differ between different databases or different ODBC drivers. Some drivers return an error when you enter a search string for Owner or Qualifier or some return no tables. If you encounter problems when entering selection criteria for Owner or Qualifier, try entering only an asterisk. If the User ID you used to access the database does not have the required access rights to some or all Tables or Views, these Tables and Views will not be displayed.

5. Select the Table(s) and View(s) you want to include in the new Dictionary Files and click OK.

**Note:** If you want to select multiple adjacent items, click the first and the last item while pressing the Shift Key.  
If you want to select multiple non-adjacent items, click the items while pressing the Ctrl Key.

The following screen is displayed for each selected Table or View.

![Screen shot of the selection criteria page](image)

Each selected Table or View will be saved in a separate Dictionary File.
6. Fill out the required fields and click OK.
   Refer to the section Description of the Available Fields (page 219) for a detailed description of each field.
   The following screen is displayed:

![Program Name](image)

7. Enter the program name and click OK.
   As this program name will also be assigned to the generated COBOL source program, you should already take the following naming rules into account:
   - The name may be from 1-8 characters in length.
   - It must begin with an alphabetic character.
   - It may contain the characters A-Z, 0-9 and embedded hyphens.
   The default name displayed is the Table or View name truncated to 8 characters and with omission of any unauthorized characters (for instance underscores).

8. The Load Delimited Options screen will be displayed.
   Define the Load Options for the selected table and click OK.

![Load Delimited Options](image)

Refer to the section Load Delimited Options (page 230) for a description of the different load options for each file type.

9. Repeat the previous steps for each selected Table or View.
   The Dictionary File for each selected Table or Views is displayed in the Tree View Window.

10. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.
11. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.

Click the Save to MetaStore icon ( ) to do so.

**Description of the Available Fields**

- **Prefix, Separator and Dictionary File Name** (page 227)
- **File Type** (page 227)
- **Key Field** (page 227)
- **File Description Size** (page 228)
- **Record Format** (page 228)
- **Block Size** (page 228)
- **Spanned** (page 228)
- **Recording Mode** (page 228)
- **Label** (page 228)
- **Column separator** (page 229)
- **Row terminator** (page 229)
- **Code-control Table Name** (page 229)
- **Database** (page 229)
- **External Source** (page 229)
- **Remarks** (page 230)

**Prefix, Separator and Dictionary File Name**

In the three fields at the top of the screen, you can enter the prefix, the separator character and the Dictionary File Name:

- The prefix will be placed in front of all Objects (file, records and fields) in order to facilitate its identification. Its maximum length is 8 characters.
- The Separator character will be placed between the defined prefix and the Dictionary File name.
- A default name for the Dictionary File is automatically filled in. The characters _F are appended to this name in order to preserve name uniqueness. You can however define another name.

The total length of prefix, separator and filename may not exceed 32 characters.

**File Type**

This read-only Field is always set to Sequential.

**Key Field**

If required, select the Key Field for the Dictionary File from the drop-down list. Fields occurring more than once are omitted from the list. The field names are listed in alphabetical order. Their names are concatenated with the names of containing fields or records.
**File Description Size**

In this field, enter the maximum record size. This corresponds to the Input-Output buffer size. The value entered must be at least the size of the largest record within the file definition.

**Record Format**

Select the required format from the drop-down list. The following options are available:

- **Fixed**: Select this option if the records of the file have all the same length. This size, expressed as a number of characters, is defined in the **File Description Size** field.
- **Undefined**: Select this option if the records in the file are no standard variable length records. Record descriptor words, which are standard for variable-length records, are not maintained at the start of each undefined record. Instead, the applications accessing the file determine the record size using other criteria. It is not possible to select this option together with the **Block Size** option. In this case, the record size defined in the following field defines the maximum number of characters contained in any record belonging to the file.
- **Variable**: Select this option if the file contains records of different sizes. The record size defined in the **File Description Size** field matches in this case the maximum record length (in number of characters)

**Block Size**

In this field, you may enter the block size. It may not be smaller than the value defined in the **File Description Size** field.

**Spanned**

Select this check box if the variable-length records may "span" two or more blocks. This option is only applicable if the Record Format is set to **Variable**, the **Block Size** is greater than the **File Description Size**, and the subtype is index, relative or sequential.

**Recording Mode**

Select the required recording mode from the drop-down list. The following options are available:

- **Native**
- **ASCII**
- **EBCDIC**

**Label**

Select the required label from the drop-down list. The following options are available:
This field contains the default Column Separator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide. You can enter another character in this field in order to override the default Column Separator for this file.

Row terminator

This field contains the default Row Terminator for Standard Files, if defined on the MetaMap Manager Settings tab of the MetaSuite INI Manager. For more information, refer to the chapter Customizing the MetaSuite INI Settings in the Install and Setup Guide. You can enter another character (sequence) in the field in order to override the default Row Terminator for this file.

Code-control Table Name

When a MetaSuite Model is generated into COBOL Source Code, this operation is performed with standard Code-control Tables. It is possible to use a customized code-control table instead, by entering its name in this field. If the table name is a three-letter word, the Code-Control parameter will be suffixed with OUT. Example: Code-control XML will become XMLOUT, and Code-control MQS will become MQSOUT. More information about code-control tables can be found in the Generator Manager Guide.

Database

Name of the database you want to feed.

External Source

Optional field. You can use it in the two following situations:

- You want to define a link between the file-name definition in MetaSuite and the physical file name on disk. The physical file name will be used automatically in the generated job.

- You want to indicate that there is no direct access to the data source. The data source might be remote, or there is need to copy or transform the source file via a simple copy or via a transformation tool. In this case.
the file, specified by the `EXTERNAL-SOURCE` parameter is the file that has to be transferred or copied to the local area. The local input file will have the name indicated by `file-name`.

This parameter will be processed in the customizable MRL Tables. By default, MetaSuite will use the following conversion rules:

<table>
<thead>
<tr>
<th>External Source Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>STD:Filename</td>
<td>This file name will be taken as input file. (This is the first method of using <code>EXTERNAL-SOURCE</code>)</td>
</tr>
<tr>
<td>FTP:Filename</td>
<td>The file will be downloaded via the FTP protocol.</td>
</tr>
<tr>
<td>CPY:Filename</td>
<td>A file copy will be done from this file to the standard file.</td>
</tr>
</tbody>
</table>

**Note:** The External Source parameter is not supported yet on every platform. UNIX and Windows are fully supported. OS390 supports the STD type. Please contact the MetaSuite support team if you want to use this option on another platform.

**Remarks**

In this field, enter your remarks to the Dictionary File. They will be stored as Business Rule in the MetaStore.

**Load Delimited Options**

This section describes the different load options for each of the Load Delimited file types.

- **ADABAS D** (page 230)
- **DB2 LUW** (page 231)
- **Informix** (page 232)
- **Oracle** (page 232)
- **Red Brick** (page 233)
- **SQL Server** (page 234)
- **Sybase** (page 235)

**ADABAS D**

The following load options are available. They are all mandatory.

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infile</td>
<td>Enter the name of the Infile.</td>
</tr>
</tbody>
</table>
| Character Set       | Select the Character Set from the drop-down list. The following options are available:  
                      - ASCII  
                      - EBCDIC |
| Fields terminated by| Enter the character to be used to terminate the fields.                  |
The following load options are available.

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save Count</td>
<td>Enter the number of physical records in the file to be loaded.</td>
</tr>
<tr>
<td>Row Count</td>
<td>In this field, specify that the load utility is to establish consistency points after the specified number of rows. This value is converted to a page count, and rounded up to intervals of the extent size.</td>
</tr>
<tr>
<td>Restart Count</td>
<td>Reserved.</td>
</tr>
<tr>
<td>Warning Count</td>
<td>Enter the number of warnings after which the load operation should be stopped. Set this parameter if no warnings are expected, but verification that the correct file and table are being used is desired.</td>
</tr>
<tr>
<td>Data Buffer</td>
<td>Enter the value to use as buffered space for transferring data within the utility.</td>
</tr>
<tr>
<td>Sort Buffer</td>
<td>Enter the value that overrides the SORTHEAP database configuration parameter during a load operation. This parameter is useful for throttling the sort memory that is used when loading tables with many indexes without changing the value of SORTHEAP, which would also affect general query processing.</td>
</tr>
<tr>
<td>Hold Quiesce</td>
<td>Select this check box if the utility should leave the table in quiesced exclusive state after the load operation.</td>
</tr>
<tr>
<td>Infile</td>
<td>This file is mandatory. Enter the name of the Infile</td>
</tr>
<tr>
<td>Message File</td>
<td>Enter the name of the Message File.</td>
</tr>
<tr>
<td>Dump File</td>
<td>Enter the name of the Dump File.</td>
</tr>
<tr>
<td>Exception Table</td>
<td>Enter the name of the exception table into which rows in error will be copied.</td>
</tr>
</tbody>
</table>
Informix

The following load options are available.

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infile</td>
<td>Mandatory field. Enter the name of the Infile</td>
</tr>
<tr>
<td>Fields Terminated By</td>
<td>Mandatory field. Enter the character that marks the end of a specific field.</td>
</tr>
</tbody>
</table>

Oracle

The following load options are available.

<table>
<thead>
<tr>
<th>Field Clause</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Options Clause - Skip</td>
<td>Enter the number of logical records to be skipped.</td>
</tr>
<tr>
<td>Options Clause - Load</td>
<td>Enter number of logical records to be loaded.</td>
</tr>
<tr>
<td>Options Clause - Errors</td>
<td>Enter the number of errors to allow on the load.</td>
</tr>
<tr>
<td>Options Clause - Rows</td>
<td>Enter the number of rows to load before a commit is issued (conventional path only). For direct path loads, rows are the number of rows to read from the data file before saving the data in the datafiles.</td>
</tr>
<tr>
<td>Options Clause - Silent - Feedback</td>
<td>Select this check box to suppress feedback errors during data load.</td>
</tr>
<tr>
<td>Options Clause - Silent - Discards</td>
<td>Select this check box to suppress discard errors during data load.</td>
</tr>
<tr>
<td>Options Clause - Silent - Errors</td>
<td>Select this check box to suppress errors during data load.</td>
</tr>
</tbody>
</table>
IKAN Solutions META SUITE META STORE MANAGER - RELEASE 8.1.3

Options Clause - Direct
Select this check box, if a direct path load must be used. Use Direct Path Loads: The conventional path loader essentially loads the data by using standard insert statements. The direct path loader (direct=true) loads directly into the Oracle data files and creates blocks in Oracle database block format.

Infile
Enter the name of the Infile.

Infile - Recoverable
Select this check box to enable the writing of the data to the redo logs. This option is available for direct path loads only.

Badfile
Enter the name of the Badfile.

Discardfile
Enter the name of the Discardfile.

Discardmax
Enter the maximum number of Discards allowed.

Load Method
Select the required Load Method from the drop-down list. The following options are available:
- APPEND
- INSERT
- REPLACE
- TRUNCATE

Fields Terminated By
Enter the character that marks the end of a specific field.

Preserve blanks
Select this check box, if you want to retain leading whitespace when optional enclosure delimiters are not present, or if you want to leave trailing whitespace intact when fields are specified with a predetermined size.

Trailing Nullcols
Select this check box, if you want any missing data to be loaded as NULL values.

Red Brick

The following load options are available.

Input Clause - Input File
Enter the name of the input file.

Input Clause - Tape Device
Enter the Tape Device parameter value.

Input Clause - Start Record
Enter the Start Record parameter value.

Input Clause - Stop Record
Enter the Stop Record parameter value.

Format Clause - Load Method
Select the required Load Method from the drop-down list. The following options are available:
- APPEND
- INSERT
- MODIFY
- REPLACE
- UPDATE
<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format Clause - Character Set</td>
<td>Select the required character set from the drop-down list. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• ASCII</td>
</tr>
<tr>
<td></td>
<td>• EBCDIC</td>
</tr>
<tr>
<td>Format Clause - Fields Terminated By</td>
<td>Mandatory field. Enter the character that marks the end of a specific field.</td>
</tr>
<tr>
<td>Discard Clause - Discardfile (ASCII)</td>
<td>Enter the name of the ASCII Discard File.</td>
</tr>
<tr>
<td>Discard Clause - Discardfile (EBCDIC)</td>
<td>Enter the name of the EBCDIC Discard File.</td>
</tr>
<tr>
<td>Discard Clause - Discardmax</td>
<td>Enter the number of discards to allow.</td>
</tr>
<tr>
<td>Discard Clause - AutoRowGen</td>
<td>Select the required option from the drop-down list. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• DEFAULT</td>
</tr>
<tr>
<td></td>
<td>• OFF: Records violating referential integrity are written to the discard file.</td>
</tr>
<tr>
<td></td>
<td>• ON: New rows are automatically generated (using the column default values) and added to the referenced tables to fully satisfy referential integrity, and the input record is added to the table being loaded.</td>
</tr>
<tr>
<td>Discard Clause - List of Tables</td>
<td>Enter the value for the List of Tables.</td>
</tr>
<tr>
<td>Discard Clause - 2nd AutoRowGen</td>
<td>Enter the value for the 2nd AutoRowGen.</td>
</tr>
<tr>
<td>Discard Clause - Additional List of Tables</td>
<td>In this field enter the value for the Additional List of Tables.</td>
</tr>
<tr>
<td>RI-DiscardFile</td>
<td>Enter the name of the RI Discard File.</td>
</tr>
<tr>
<td>Optimize</td>
<td>Select the required option from the drop-down list. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• OFF</td>
</tr>
<tr>
<td></td>
<td>• ON</td>
</tr>
<tr>
<td>Duplicates Discardfile</td>
<td>Enter the value of the corresponding parameter.</td>
</tr>
<tr>
<td>Remarks</td>
<td>Optional field. Enter user comments.</td>
</tr>
</tbody>
</table>

**SQL Server**

The following load options are available:
The following load options are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields Terminated By</td>
<td>Mandatory field. Enter the character terminating the individual fields.</td>
</tr>
<tr>
<td>Row Terminator</td>
<td>Mandatory field. Enter the character string marking the end of a row.</td>
</tr>
</tbody>
</table>

**Sybase**

The following load options are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields Terminated By</td>
<td>Mandatory field. Enter the character terminating the individual fields.</td>
</tr>
<tr>
<td>Row Terminator</td>
<td>Mandatory field. Enter the character string marking the end of a row.</td>
</tr>
</tbody>
</table>

### 18.3. Load RDBMS

Supported File Types:
- Catalog
- DB2 DDL

**Procedures**
- [Catalog](#) (page 236)
- [SQL DDL](#) (page 238)
Catalog

1. On the Collect File screen, open the Target tab and select RDBMS as File Type.

2. Select the required format from the drop-down list and click OK. The RDBMS Logon screen is displayed.

3. Fill out the required fields and click OK.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Enter your User Name to get access to the database. The User ID displayed in this field is the default User ID defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter your password.</td>
</tr>
<tr>
<td>Data Source Name</td>
<td>Enter the required DSN. The DSN displayed in this field is the default DSN defined in Metasuite.ini.</td>
</tr>
<tr>
<td>Database</td>
<td>Enter the name of the database.</td>
</tr>
</tbody>
</table>

**Note:** If required, make the necessary selections to obtain the ODBC connection.

Once the ODBC connection is established, the following screen is displayed.
4. In the upper part of the screen, define your selection criteria and click Show. The lower part of the screen will list the Tables and/or Views matching your selection criteria.

<table>
<thead>
<tr>
<th>Selection Criterion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a Table or View name in this field, if you want to limit your search to tables or views with this name. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>Enter a Qualifier in this field, if you want to limit your search to tables or views with this Qualifier. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
<tr>
<td>Owner</td>
<td>Enter a User ID in this field, if you want to limit your search to tables or views owned by this User ID. You may use the asterisk (*) and the question mark (?) as standard wildcard characters.</td>
</tr>
</tbody>
</table>
| Type                | Select one of the following types:  
  • TABLE: include database tables in your search.  
  • VIEW: include database views in your search.  
  • BOTH: include both database tables and views in your search. |

**Note:** Owner and Qualifier are ODBC terms. The meaning of these terms can differ between different databases or different ODBC drivers. Some drivers return an error when you enter a search string for Owner or Qualifier or some return no tables. If you encounter problems when entering selection criteria for Owner or Qualifier, try entering only an asterisk. If the User ID you used to access the database does not have the required access rights to some or all Tables or Views, these Tables and Views will not be displayed.

5. Select the Table(s) and View(s) you want to include in the new Dictionary Files and click OK.

**Note:** If you want to select multiple adjacent items, click the first and the last item while pressing the **Shift** Key.  
If you want to select multiple non-adjacent items, click the items while pressing the **Ctrl** Key.  
The **RDBMS File Information** screen is displayed.
6. Fill out the required fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name <em>(Enter the filename for ADD FILE statement in MDL-file)</em></td>
<td>Enter the name of the Dictionary File</td>
</tr>
<tr>
<td>RDBMS SubType Information</td>
<td>Select the required SubType from the drop-down list. You can also select Generator Default to use the SQL dialect set as default in the Generator Manager.</td>
</tr>
<tr>
<td>Uncommitted Read <em>(DB2 only)</em></td>
<td>(DB2 only) Select this check box, if you want to perform a so-called dirty read. This means reading data without checking if the data has been locked or not.</td>
</tr>
<tr>
<td>Fetch First Row Only <em>(DB2 only)</em></td>
<td>(DB2 only) Select this check box, if you want to read only the first row, if different rows contain the same key.</td>
</tr>
</tbody>
</table>

The Dictionary File for each selected Table or Views is displayed in the Tree View Window.

7. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

8. Once the settings of the Dictionary Files match your requirements, you can save them to the MetaStore.
   Click the Save to MetaStore icon ( ) to do so.

**SQL DDL**

1. On the Collect File screen, open the Target tab and select RDBMS as File Type.

2. Select the required format and click OK.
   The default folder (as defined in the MetaSuite.ini file) opens in a new window and displays the available SQL DDLs (extension .ddl).

   **Note:** If the DDL you need is not available in this folder, browse to the required folder.

3. Select the required DDL and click Open.
   The SQL DDL File Information screen is displayed.
4. Fill out the required fields and click OK.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Enter the name of the Dictionary File. The name of the DDL file is used as default value.</td>
</tr>
<tr>
<td>RDBMS Subtype Information</td>
<td>Select the required SubType from the drop-down list. The following options are available:</td>
</tr>
<tr>
<td></td>
<td>• DB2 for OS/400</td>
</tr>
<tr>
<td></td>
<td>• DB2 for z/OS</td>
</tr>
<tr>
<td></td>
<td>• DB2 LUW</td>
</tr>
<tr>
<td></td>
<td>• DB2/2</td>
</tr>
<tr>
<td></td>
<td>• DB2_VSE</td>
</tr>
<tr>
<td></td>
<td>• Generator Default</td>
</tr>
<tr>
<td></td>
<td>• Informix</td>
</tr>
<tr>
<td></td>
<td>• Ingres</td>
</tr>
<tr>
<td></td>
<td>• MySQL</td>
</tr>
<tr>
<td></td>
<td>• ODBC</td>
</tr>
<tr>
<td></td>
<td>• Oracle</td>
</tr>
<tr>
<td></td>
<td>• Oracle/RDB</td>
</tr>
<tr>
<td></td>
<td>• SESAM</td>
</tr>
<tr>
<td></td>
<td>• SQLServer</td>
</tr>
<tr>
<td></td>
<td>• Sybase</td>
</tr>
<tr>
<td></td>
<td>• Teradata</td>
</tr>
</tbody>
</table>

The following screen is displayed.

5. In the upper part of the screen, define your selection criteria and click Show.
   The lower part of the screen will list the Tables and/or Views matching your selection criteria.
6. Select the Table(s) and View(s) you want to include in the new Dictionary Files and click **OK**.

**Note:** If you want to select multiple adjacent items, click the first and the last item while pressing the **Shift** Key. If you want to select multiple non-adjacent items, click the items while pressing the **Ctrl** Key.

7. If required, edit the definitions.

The procedure to edit the definitions is identical to building Dictionary Files manually. See [Building Dictionary Files Manually - Overview](on page 20).

8. Once the settings of the Dictionary File match your requirements, you can save it to the MetaStore.

Click the **Save to MetaStore** icon ( ) to do so.
CHAPTER 19

Importing MDL Files

Importing an MDL File means adding a Dictionary File to the MetaStore. It can be useful to import MDL files, if:

- you need to transfer a Dictionary File from one to another MetaStore (e.g. if you have separate MetaStore for testing and actual use)
- you need to transfer a Dictionary File to a newer version of the MetaStore.

1. On the MetaStore Toolbar, click the Import File icon ( ).
   The default folder (as defined in the MetaSuite.ini file) opens in a new window and displays the available MDL files.

   **Note:** In order to access this screen, you can also select the Import File option accessible from the File menu or from the MetaStore context menu in the Tree View Window.

2. Select the required MDL file (or browse to another folder first, if it is not available in this one) and click Open.
   The contents of the MDL file is displayed in the Tree View Window.

3. If required, edit the definitions.
   The procedure to edit the definitions is identical to building Dictionary Files manually. See Building Dictionary Files Manually - Overview on page 20.

4. Once the settings of the Dictionary File match your requirements, you can save it to the MetaStore.
   Click the Save to MetaStore icon ( ) to do so.
Exporting a Dictionary File means generating a text file containing the definition of the selected Dictionary File(s) to one of the supported formats:

<table>
<thead>
<tr>
<th>Format</th>
<th>Use</th>
</tr>
</thead>
</table>
| MDL (MetaSuite Definition Language) | • Transfer a Dictionary File from one MetaStore to another (e.g. if you have separate MetaStores for testing and actual use)  
• Edit the MDL code directly. Refer to the section Definition Language Commands (page 258).  
Note: This use is no longer advised, as all required editing can be performed using the MetaStore Manager GUI. |
| MXL             | Conversion from an original format into Delimited (without using the MetaMap Manager). |
| XML             | Conversion from an original format into an XML format.             |
| XMI             | Conversion from an original format into an XMI format, a standard format for modeling tools. |
| OWB             | Will be supported in the next release.                             |

20.1. General Export Procedure

1. Open the Dictionary File(s) to be exported.  
To do so, click the Open File icon ( ) from the MetaStore Toolbar. Select the required Dictionary File(s) from the list and click Open.

2. Select the Dictionary File to be exported in the Tree View Window.

Note: You can also export all opened Dictionary Files by selecting the MetaStore icon at the top of the Tree View Window.

3. From the File menu, select Export. Then choose the required format from the submenu.

Note: If you want to export the Dictionary File(s) to MDL, you can also click the Export File icon ( ) on the MetaStore Toolbar.
The default folder (as defined in the MetaSuite.ini file) opens in a new window.

4. Confirm or enter the name and location of the exported File.
   - If you selected an individual Dictionary File, the default name for the exported file is the identical. If you selected the MetaStore icon, the default name for the exported MDL file is *DATA*.
   - The extension reflects the selected format.
   - The suggested location is determined by the MetaSuite initial settings. For more information, refer to the chapter *Customizing the MetaSuite INI Settings* in the *Installation and Setup Guide*.

5. Click Save.
   The file is exported.
CHAPTER 21

Selecting another User Profile

In MetaSuite, a *User Profile* is a combination of personalized settings which are saved in an INI file. The default name of this INI file is `MetaSuite.ini` and it is located in the user's `AppData\Roaming\MetaSuite` folder. The initial settings were defined during the MetaSuite installation. It is possible to change the settings in the `MetaSuite.ini` file or to create additional INI files with different settings.

The option *User Profile* from the *Tools* menu allows to select another User Profile or to reload a modified User Profile, so that its settings become active.

**Note:** The procedure on how to update User Profiles is explained in the *INI Manager Guide*.

1. Select *Tools > User Profile*.
   All available INI files are displayed.

2. Select the required INI file and click *Open*. 
It is possible to save multiple versions of a Dictionary File. The versions are saved in a Source Control system like Microsoft SourceSafe and can be retrieved as files written in the MDL (*MetaSuite Definition Language*) format.

**22.1. Establishing the Connection Between MetaStore and the Source Control System**

When you start a new MetaStore Manager session, the connection to the Source Control system is not automatically established. If you want to use the Source Control, you need to establish the connection manually.

1. Select *Source Control > Connect to Source Control...*  
   A screen similar to this one is displayed:

2. Fill out the fields as required and click *OK.*

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
<td>Enter a username you can use to access the Source Control database.</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the required password.</td>
</tr>
</tbody>
</table>
| Database     | Enter the name of the Source Control database to be used.  
                You can also use the *Browse button* to access the required database. |
The following screen is displayed:

![Add to SourceSafe dialog box](image)

3. Select MetaSuite > MSP and click OK.

**Note:** The MSP folder is the default folder for MetaSuite Packages. You can modify the default folder using the INI Manager (refer to the INI Manager User Guide for more information).

The following message is displayed in the Output Window: *Source code-control: Connect to project was successful.*

### 22.2. Terminating the Connection Between MetaStore and the Source Control System

If you do not want to work with Source Control any longer, you can terminate the connection.

1. Select **Source Control > Disconnect from Source Control...**

   **Results:**
   - The following message is displayed in the Message window: *Source code control: Disconnect was successful*
   - The special Dictionary File icons ( and ) are replaced by the standard Dictionary File icon ( ).

   **Note:** When you make changes to Dictionary Files while the connection to Source Control is inactive, these changes will NOT be taken into account by the Source Control database. As a result, a discrepancy will occur between the MetaStore and the Source Control database.

### 22.3. Adding Dictionary Files to Source Control

The purpose of adding Dictionary Files to Source Control is to save multiple versions of these files and to be able to retrieve each of these versions.
1. Create or open the Dictionary File you want to add to Source Control. Refer to Building Dictionary Files Manually - Overview (page 20).

**Note:** You can verify if a Dictionary File has already been added to Source Control by selecting Show Status from the Source Control menu. See Showing the Source Control Status of Opened Source Files on page 247.

2. Right-click the Dictionary File name in the Tree View Window and select Add to Source Control. The following screen is displayed:

3. Enter a comment and click OK.

The Dictionary File is now checked in. In the Tree View Window, checked-in Dictionary Files are represented by the icon.

If you want to make changes to a checked-in file, you first have to check out the file. See Performing Changes to Dictionary Files Under Source Control on page 248.

### 22.4. Showing the Source Control Status of Opened Source Files

1. Open the Dictionary File you want to display the status for.

2. Select Show Status from the Source Control menu. The applicable status message is displayed. The following table lists the different possibilities:

<table>
<thead>
<tr>
<th>Status</th>
<th>Displayed message</th>
<th>Icon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dictionary File not added to Source Control</td>
<td>![Status message for Dictionary File not added to Source Control]</td>
<td></td>
</tr>
<tr>
<td>Dictionary File added to Source Control, not checked out</td>
<td>![Status message for Dictionary File added to Source Control, not checked out]</td>
<td></td>
</tr>
</tbody>
</table>
22.5. Performing Changes to Dictionary Files Under Source Control

1. Make the required Dictionary File version available in the Tree View Window.

2. Right-click the Dictionary File name and select **Check Out**.
   
   The following screen is displayed:

3. If required, enter a Comment and click **OK**.
   The Comment will be available in the Source Control program (Visual SourceSafe, ClearCase, etc.)

   The Dictionary File is checked out. Its icon in the Tree View Window changes to [valid].

4. If the selected Dictionary File was already checked out by another user, a warning is displayed. Click **Yes** to check out the file as well.
   
   When you and the other user check in the Dictionary File, they will both get a Version Number, and they will be both managed by Source Control.

5. Perform the required changes to the Dictionary File.
   
   See **Performing Changes to Dictionary Files Under Source Control** on page 248.
6. Once the required changes have been performed, select Check In from the Source Control menu.
   
The following screen is displayed:

![Check In from source control](image_url)

7. If required, enter a Comment and click OK.
   
The Comment will be available in the Source Control program (Visual SourceSafe, ClearCase, etc.)

22.6. Undoing the Check-Out of a Dictionary File

You can undo the check-out of a Dictionary File, if you want to revert to the last saved version of a Dictionary File while ignoring the changes made in the mean time.

1. Select the required Dictionary File in the Tree View Window.

2. Select Undo Check Out from the Source Control menu.
   
The following message appears:

![Undo Check Out message](image_url)

3. Click Yes to confirm the operation.

   The check-out is undone and the Dictionary File icon on the Tree View Window changes to .
   You can also click No to cancel the operation and keep the file checked out.
Version Management with Source Control (page 245) offers external version management: it maintains multiple versions of metadata in another database rather than in the MetaStore.

### 23.1. Metadata Versioning Rules

The MetaStore can contain several versions of a Dictionary File. The file type and name are identical, but the version number is different.

It is not possible to have several Dictionary Files with the same name, but different file types.

If you create or collect a new Dictionary File, the Version field on the Dictionary File properties panel should be defined as follows:

- If a Dictionary File with the same name already exists, enter the highest version number + 1 in this field.
- If a Dictionary File with the same name does not yet exist, enter 1 in this field.

When you import an MDL file, the Version should be set to the `nnnn` version number on the `ADD FILE` line of the file. If the version information is not present, the Version field on the Dictionary File properties panel should be defined as follows:

- If a Dictionary File with the same name already exists, enter the highest version number + 1 in this field.
- If a Dictionary File with the same name does not yet exist, enter 1 in this field.

When you export an MDL file, the Version is used to set the `nnnn` version number on the `ADD FILE` line of the file. If Version field is not available, the `nnnn` version number is set to 1.

The version number should consist of 1 to 4 digits, in the range between 1 and 9999. If the version number is 9999, the next build number is set to 1 again.

If a file without version indication is opened, the version will be set to 1.
23.2. Incrementing the Version Number of a Dictionary File

1. Open a Dictionary File in MetaStore Manager.

2. Perform the required changes.


4. Save the Dictionary File to the MetaStore.
   The following screen is displayed:

5. Click Yes.
   The incremented version is saved in the MetaStore.
23.3. Creating a New Instance of a Dictionary File

1. Open the Dictionary File you want to duplicate.

2. Right-click the Dictionary File name and select Create New Version.
   A duplicate of the selected Dictionary File will be added at the bottom of the Tree View Window.

   **Note:** The version number has automatically been incremented by 1.
   If required, you can modify this version number. See Incrementing the Version Number of a Dictionary File on page 251.

3. Select the newly created instance and save the Dictionary File to the MetaStore.
   The following screen is displayed:

4. Click Yes.
   The new instance is saved in the MetaStore.

23.4. Opening a Specific Version of a Dictionary File

1. Click the Open/Delete File ... icon on the MetaStore Toolbar.
   A screen similar to this one is displayed:

   If there are multiple versions for a Dictionary File, these are indicated in the column Available Versions.
2. Select the Dictionary File with multiple versions and click the *Show History* button.
   The following screen is displayed:

3. Select the required version(s) and click *Open*.
   The selected version(s) are displayed in the Tree View Window.
CHAPTER 24

Calling the MetaStore Manager in Batch

Users can call the MetaStore Manager in batch in order to
- import new definitions into the MetaStore dictionary
- export existing Metadata from the MetaStore dictionary to an MDL file.

The `MSBSTORE.exe` program (for MetaSuite Batch MetaStore) is located in the MetaSuite installation folder. The MDL file may contain multiple File Definitions. It can be used with an MS-DOS prompt or in batch mode.

24.1. Using MSBSTORE to Import MDL Files into the MetaStore Dictionary

Command format

```
msbstore -v[:version | +] [options] MDL_Filename
```

Where `MDL_Filename` is the name of the file containing the data description in MDL format.

The following table contains an overview of what version number is assigned to the Dictionary File in specific situation:

<table>
<thead>
<tr>
<th>Option</th>
<th>MDL version number</th>
<th>MetaStore version number</th>
<th>New version number</th>
</tr>
</thead>
<tbody>
<tr>
<td>-v</td>
<td>No version specified</td>
<td>Dictionary File not yet in MetaStore</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>No version specified</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>Dictionary File not yet in MetaStore</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>x (in all cases)</td>
</tr>
<tr>
<td>-v+</td>
<td>No version specified</td>
<td>Dictionary File not yet in MetaStore</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>No version specified</td>
<td>y</td>
<td>y+1</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>Dictionary File not yet in MetaStore</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>y</td>
<td>y+1, if x&lt;=y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>x, if x&gt;y</td>
</tr>
</tbody>
</table>

For more information on the available options, refer to [Optional Parameters Overview](page 256).
Example

```plaintext
msbstore.exe -v -i:MetaSuite.ini "%MDLDIR%\XML-INP-PAIN.MDL"
Metasuite MetaStore Manager Batch Tool Version 08.01.02 Build 999
No path information specified FOR INI file, assuming C:\Documents and Settings\fib\Application Data\MetaSuite.
Logging on as 'Metasuit' to database 'Metasuit' at server 'MetaStore:813:MsAccess' ...
Caching completed. The dictionary 'MetaSuite MetaStore' contains 57 files (versions included).
Importing ... : Z:\UPLOAD\CSDC\MetaSuite\TEST FILES\STANDARD MDL FILES\XML-INP-PAIN.MDL
Importing ... completed with 0 warnings, and 0 errors.
Save XML-PAIN (StandardFile) ...
Save MetaSuite MetaStore (Dictionary) ...
Batch command(s) succeeded.
```

24.2. Using MSBSTORE to Export MDL Files from the MetaStore Dictionary

Command format

```plaintext
msbstore -e[:version] [options] MetaStore_Filename
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-e[:version]</td>
<td>Export the specified MDL file from the MetaStore. If no version is specified in the MDL file, the latest version will be taken.</td>
</tr>
<tr>
<td>MetaStore_Filename</td>
<td>The name of the file that will be exported. If * is used as filename, all existing files in the MetaStore will be exported to DATA.MDL.</td>
</tr>
</tbody>
</table>

For more information on the available options, refer to Optional Parameters Overview (page 256).

Example

```plaintext
D:\Program Files\IKAN Solutions\MetaSuite813>msbstore -e -i:metasuite.ini -f:employee-master?
Metasuite MetaStore Manager Batch Tool Version 08.01.02 Build 267?
No path information specified FOR INI file, assuming C:\Documents and Settings\fib\Application Data\MetaSuite.?
Logging on as 'Metasuit' to database 'Metasuit' at server 'MetaStore:813:MsAccess' ...
Caching completed. The dictionary 'MetaSuite MetaStore' contains 60 files (versions included).?
Exporting "employee-master (v 1)". ...?
Export of employee-master to D:\Program Files\IKAN Solutions\MetaSuite813\MDL\employee-master.Mdl: Done.?
Batch command(s) succeeded.?
```
24.3. Using MSBSTORE to Collect Files

Command format

```
msbstore -c[++][-t:collectType [options]] -f:Collect_filename
```

Where *filename* is the name of the collected file.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>Collect the specified file into the MetaStore. If no version is specified in the collected file this is the same as -c+.</td>
</tr>
<tr>
<td>-c+</td>
<td>Collect the specified file into the MetaStore and create a new version of the files.</td>
</tr>
</tbody>
</table>
| -t:collectType | Collect type:  
cbl - COBOL copy book  
pli - PL/I include Book  
rp - Idms record punch  
sp - Idms schema punch  
ddl - SQL DDL or DB2 DDL  
sap - SAP/R3DMI 3.1, SAP/R3DMI 4.0  
sxd - XSD |
| -f:Collect_filename | The name of the collected file. |

For more information on the optional parameters, refer to Optional Parameters Overview (page 256).

Example

D:\Program Files\IKAN Solutions\MetaSuite813>msbstore -c -i:metasuite.ini -f:employee-master.cbl -t:cbl  
Metasuite MetaStore Manager Batch Tool Version 08.01.02 Build 267  
No path information specified FOR INI file, assuming C:\Documents and Settings\Application Data\MetaSuite.  
Logging on as 'Metasuit' to database 'Metasuit' at server 'MetaStore:813:MsAccess'  
Caching completed. The dictionary 'MetaSuite MetaStore' contains 60 files (versions included).  
Started collecting of 'Z:\UPLOAD\CSDC\MetaSuite\TEST FILES\MISC COLLECT FILES\employee-master.cbl'.  
Save employee-master (StandardFile) ...  
Save employee-master (StandardFile) ... OK  
Batch command(s) completed with error(s).

24.4. Optional Parameters Overview

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h or -?</td>
<td>Help. Displays the options.</td>
</tr>
</tbody>
</table>
24.5. MSBSTORE Return Codes

There are two possible Return Codes after an MSBSTORE run:

- 0 (zero): Import or export successful. A message is written to stdout.
- 4: Import or export failed. An appropriate message is written to stdout and to the `msbstore.log` file. This file is located in the MetaSuite installation folder.

24.6. Calling MetaStore Manager Via the Commandline

MetaStore Manager can be called by means of the following command:

```
MetaStore [filename1] [filenameN] <Ins>
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i:file name</td>
<td>INI file with preset settings for MetaSuite. Mandatory if no user, pass, etc. are supplied. INI settings will be overridden by the command line settings.</td>
</tr>
<tr>
<td>u:userid</td>
<td>User ID to log on to the MetaStore.</td>
</tr>
<tr>
<td>-p:password</td>
<td>Password to log on to the MetaStore.</td>
</tr>
<tr>
<td>-s:DSN</td>
<td>Data source name for ODBC connection to the MetaStore.</td>
</tr>
<tr>
<td>-d:database</td>
<td>Database to log on to the MetaStore.</td>
</tr>
<tr>
<td>-o:owner</td>
<td>The owner of the MetaStore tables.</td>
</tr>
<tr>
<td>-l:folder</td>
<td>The folder where the MDL:</td>
</tr>
<tr>
<td></td>
<td>• will be placed (optional) - for export</td>
</tr>
<tr>
<td></td>
<td>• resides (optional) - for import</td>
</tr>
</tbody>
</table>

filename1 ... filenameN  optional  These files can be collectable files or MDL files. Metastore will investigate the extension of the file name and will collect or import the file accordingly.

Ins  mandatory  The installation directory specified during installation.
25.1. ADD FILE

The ADD FILE command is used to add a file definition. This command defines the physical characteristics of a file. The ADD FILE command options are used to specify the type of file organization, the size and format of the records in the file, the recording mode used to store file data, the type of label records used, and the name of the field used as a key for random access.

Format

ADD FILE file-name
   [VERSION file-version]
   [TYPE file-type]
       {FIXED | VARIABLE | UNDEFINED} record-size
   [BLOCK block-size] [SPANDED]
   [MODE {EBCDIC | ASCII}] [LABEL {STANDARD | OMITTED}]
   [KEY (key-field)] [CODE-CONTROL table-name]
   [FETCH-FIRST-ONLY] [UNCOMMITTED-READ]
   [EXTERNAL-SOURCE 'external-source-name']
   [COLUMN-SEPARATOR 'column separator characters']
   [ROW-TERMINATOR 'row terminator characters']
   [CCSID coded-character-set-identifier]
   [XPATH XML-path]
   [XML-DECLARATION xml-declaration]
   [RULE business-rule] [NOTE note]

ADD FILE

Required.

File-name is the name of the file being defined.

The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters A-Z and 0-9.
- It may contain the following embedded special characters: $, #, @, _ (underscore) and - (hyphen).
- File names must be unique in the MetaSuite Generator Library.

The example below defines two files: PAYROLL-DETAIL, with a fixed record length of 100 and EMPLOYEE-MASTER, an indexed sequential file with a fixed record length of 150.
ADD FILE PAYROLL-DETAIL FIXED 100 -
RULE CONTAINS DETAIL PAY INFORMATION FOR EACH EMPLOYEE FOR EACH PAY-PERIOD
ADD FILE EMPLOYEE-MASTER TYPE INDEX FIXED 150 KEY (EMPLOYEE-NUMBER)

Some restrictions apply when defining certain types of database files. File naming considerations for each type of supported database file are described in the supplement appropriate for the database.

VERSION
Optional.
The VERSION option specifies the version-number of a file.
The following command defines a sequential file with version 2:

ADD FILE PAYROLL-DETAIL VERSION 2 TYPE SEQUENTIAL FIXED 100

TYPE
Optional.
The TYPE option specifies the type of file organization used by the file. If omitted, it defaults to TYPE SEQUENTIAL (indicating a sequential file).

TYPE {FUNCTION | SEQUENTIAL | INDEX | RELATIVE | VSAM | SEQ-LINE | SEQ-RECORD | DELIMITED | XML | database}
Each type of file organization is described separately.

| FUNCTION | TYPE FUNCTION indicates that the records of the file describe the input and/or output fields required for a subroutine INVOKE command. The following command defines a function file named CALLING-PARMS: ADD FILE CALLING-PARMS TYPE FUNCTION |
| SEQUENTIAL | TYPE SEQUENTIAL, the default file type, indicates that the records of the file are physically stored in sequential order and may only be retrieved in the order in which they were written. The following command defines a sequential file named PAYROLL-DETAIL: ADD FILE PAYROLL-DETAIL TYPE SEQUENTIAL FIXED 100 This type corresponds with Line Sequential on LUW systems and with Record Sequential on Mainframe systems. Note that SEQUENTIAL describes any sequential file, regardless of the media (disk, tape, card, etc.) on which it is stored. |
| INDEX | TYPE INDEX indicates that the records of the file are sequenced logically (but not necessarily physically) according to the value of a specified key-field. Records of this type of a file may be accessed sequentially in key sequence, or randomly, for specific values of the key-field. The key-field for a TYPE INDEX file must be identified using the KEY option (described later). In IBM terminology, this type of file organization is referred to as ISAM: an acronym for the Indexed Sequential Access Method. The following command defines an ISAM employee master file: ADD FILE EMPLOYEE-MASTER TYPE INDEX FIXED 150 KEY (EMPLOYEE-NUMBER) Note: Before referencing this file in a program request, the key-field (EMPLOYEE-NUMBER) must be defined as a field within a record of the EMPLOYEE-MASTER file. |
### RELATIVE

**TYPE RELATIVE** indicates that the file is a relative record file in which records are stored and retrieved according to the value of a relative record number (the number of the record relative to the beginning of the file). A file with relative file organization is referred to as a BDAM (Basic Direct Access Method) file.

The following ADD FILE command defines a TYPE RELATIVE file named STATE-TABLES:

```
ADD FILE STATE-TABLES TYPE RELATIVE FIXED 522
```

**Note:** Unlike other keyed file types (ISAM and VSAM KSDS), TYPE RELATIVE files are not defined using the KEY option to identify the field within the record that contains the random access key. This is because the *key-fields* for a relative file are not actually contained within the records, but are rather maintained by the BDAM access method externally to the records. MetaSuite will automatically identify, define and maintain relative-record access *key-fields*.

### VSAM

**TYPE VSAM** indicates that the IBM’s Virtual Storage Access Method is used to store and retrieve the records of the file. There are two types of VSAM files supported by the system: ESDS (Entry Sequence Data Set), and KSDS (Key Sequence Data Set).

ESDS files are analogous to sequential files, and KSDS files are analogous to indexed sequential files. Like TYPE INDEX files, TYPE VSAM files of the KSDS variety must be defined with the KEY option (described below), to identify the field on whose value the sequencing of the file is based.

The following commands illustrate the definition of ESDS and KSDS type VSAM files:

```
ADD FILE PAYROLL-DETAIL TYPE VSAM FIXED 100
ADD FILE EMPLOYEE-MASTER TYPE VSAM FIXED 100 KEY (EMPL-NUMBER)
```

Before referencing the second file above (EMPLOYEE-MASTER) in a program request, the *key-field* (EMPL-NUMBER) must be defined as a field within a record of the EMPLOYEE-MASTER file.

In addition to the primary *key-field* for a VSAM KSDS file, secondary index fields may exist. These fields provide alternate access paths to the records of the file. For example, the primary *key-field* of an employee master file might be the employee number, meaning that the records of the file are organized logically and accessed (randomly or in sequence) by the employee number. If it is frequently necessary to access the records of this file in some other sequence (employee name, for example), a secondary index field would be defined (to VSAM) to provide an alternate access path. The records of the file could then be accessed in sequence by either employee number or employee name.

A secondary index path is defined as a separate VSAM file, with the name of the secondary index field being used in the KEY specification (instead of the primary index field).

For example, the following two commands define the same physical file. However, when the first file definition is accessed in a program request, the records will be returned in EMPLOYEE-NUMBER sequence. When the second file definition is accessed, the records will be returned in EMPLOYEE-NAME sequence.

```
ADD FILE EMPLOYEE-BY-NUMBER TYPE VSAM FIXED 150 KEY (EMPLOYEE-NUMBER)
ADD FILE EMPLOYEE-BY-NAME TYPE VSAM FIXED 150 KEY (EMPLOYEE-NAME)
```

### SEQ-LINE

**Type SEQ-LINE** indicates that the file is defined as line sequential.

```
ADD FILE employee TYPE SEQ-LINE FIXED 204 LABEL STANDARD
```

### SEQ-RECORD

**Type SEQ-RECORD** indicates that the file is defined as record sequential.

```
ADD FILE employee TYPE SEQ-RECORD FIXED 204 LABEL STANDARD
```
Record-size

Record-size defines the type of the records of the file and their (maximum) length. When coded, one of the three formats (described individually below) must be used.

**FIXED record-size**

FIXED indicates that the records of the file are all the same number of characters in length. Record-size defines the character length of the records. As an example, the following command defines a file whose records are all 100 characters in length:

```cobol
ADD FILE PAYROLL-DETAIL FIXED 100
```

**VARIABLE record-size**

VARIABLE indicates that records of more than one length exist on the file, with record-size specifying the maximum number of characters contained in any one record (exclusive of the four-character "record descriptor word" that is always present at the beginning of a variable length record). The following command defines a file containing variable length records having a maximum length of 482 characters:

```cobol
ADD FILE ACCOUNT-TRANSACTIONS VARIABLE 482
```

**Note:** When defining the fields of the records of variable-length record files, the first data field begins in POSITION 1; the four-character "record descriptor word" that is present at the start of each record cannot be accessed directly from the generated COBOL record description.
UNDEFINED record-size

**UNDEFINED** indicates that the record length is variable, but that the records are not standard variable length records. (Record descriptor words, which are standard for variable-length records, are not maintained at the start of each UNDEFINED record; rather, the application programs accessing the file determine the record size according to other criteria.)

When the UNDEFINED option is coded, the BLOCK option (described below) may not be coded. **Record-size** specifies the maximum number of characters contained on any record in the file.

The following command illustrates the definition of a file containing records of an undefined format.

```
ADD FILE FOREIGN-DATA UNDEFINED 4095
```

### BLOCK

Optional.

**Block-size** specifies the maximum number of characters contained in a block of records in the file. For variable-length records, the block size includes the four-byte record descriptor word for each record. If the BLOCK option is omitted, block size defaults to the same size as the record size.

The **block-size** is not allowed in conjunction with the UNDEFINED option (described above).

For sequential files in the z/OS environment, the block size defined is for documentation purposes only; generated programs accessing such files will determine the block size at execution time, either from the file label or from the JCL. In the VSE environment, this option must be coded in all cases where the block size is not equal to the record size.

The following commands illustrate the use of the BLOCK option to define the maximum block size of a file:

```
ADD FILE PAYROLL-DETAIL FIXED 100 BLOCK 3000
ADD FILE SMF-EXTRACT VARIABLE 6628 BLOCK 32760
```

### SPANNED

Optional.

Applicable only when both the VARIABLE and BLOCK options (described above) are coded. **SPANNED** indicates that each variable length record may "span" two or more blocks.

The command below illustrates the use of the **SPANNED** option:

```
ADD FILE SALES-HIST VARIABLE 4080 BLOCK 2084 SPANNED
```

### MODE

Optional.

**MODE** defines the recording mode of the file: EBCDIC (Extended Binary Coded Decimal Interchange Code) or ASCII (American National Standard Code for Information Interchange). The default recording mode depends on the generator that is used. The following command illustrates the use of the MODE option to define a file containing ASCII data:

```
ADD FILE TAX-TABLES FIXED 800 BLOCK 4000 MODE ASCII
```

**Note:** The OS/VS COBOL parameter LANGLVL(1) must be included in the COBOL parameter list for any program that accesses an ASCII file.
LABEL

Optional.

*LABEL* defines whether the file has standard label records or no label at all.

*STANDARD*, the default label-type, indicates that the file has standard label records (as defined by IBM). A file label contains information such as the creation date and the length of the file. *OMITTED* indicates that the file does not contain label records. The following command illustrates the use of the *LABEL* option to define a file containing no label records:

```
ADD FILE PAY-HIST FIXED 266 BLOCK 2660 LABEL OMITTED
```

KEY

Optional.

Applicable for type INDEX, type VSAM (KSDS), sequential and SQL arrays. The *KEY* option names the field (*key-field*) whose value may be used to access a record in the file directly or identify the external array search *key-field*. Before the records of the file can be accessed randomly or searched for in a generated program, *key-field* must be defined as a field within a record of *file-name*.

The following command identifies the *EMPLOYEE-NUMBER* field as the random access *key-field* of the VS-EMPLOYEE-MASTER file:

```
ADD FILE VS-EMPLOYEE-MASTER TYPE VSAM FIXED KEY (EMPLOYEE-NUMBER)
```

CODE-CONTROL

Optional.

The *CODE-CONTROL* option allows you to override the normal system generated COBOL code used to process this file, as directed by the MetaSuite Generator library table identified by *table-name*. This is used when you want to invoke an external I/O module to read or write the file or to expand or compress data. When this option is used, the table specified by *table-name* must exist in the MetaSuite Generator library before programs using the file can be generated.

If the *table-name* is a three letter word than the *CODE-CONTROL* parameter will be suffixed with "INP".

Example: Code-control XML will become XMLINP, and Code-control MQS will become MQSINP.

FETCH-FIRST-ONLY

Optional.

Use this option to include FETCH FIRST (1) ROW ONLY in select statements on this source file.

The FETCH FIRST (1) ROW ONLY clause was implemented in V7 of DB2 for z/OS. The major benefit of this query is that it limits the number of rows returned by the query, regardless of how many rows qualify from the WHERE clause.

Remark: Older compilers versions may not support this option, so be careful when using this feature.

UNCOMMITTED-READ

Optional.

Uncommitted Read (UR) is also known as "dirty read" or "read through locks".

Use this option to specify WITH UR in select statements on this source file.

This option will override the isolation level with which the plan or package was bound.
UR requires Type 2 indexes if an index access path is to be used. Please note that using UR can produce strange results. Use it only if you are sure it can do no damage.

**Note:** Contrary to popular belief (to coin a phrase) queries using UR do acquire locks, though not the kind that are likely to cause contention. First, they acquire a special "mass delete" lock in share (S) mode on the target table or table space; this is not the same as a normal share (S) lock on a table or table space. The special mass delete S lock prevents any other process from issuing a mass delete (a delete statement without a WHERE clause) while the query is running. Thus, it is possible to have contention between a UR query and a mass delete. Second, UR queries acquire IX locks on any table space they may happen to use in the temporary work file database. This lock prevents the work file table space from being dropped while the query is running; it is of no concern to the developer because the work file table spaces are dropped, if at all, only during DB2 subsystem maintenance, and no queries would be running while such maintenance was in progress.

Remark: Older compilers versions may not support this option, so be careful when using this feature.

**EXTERNAL-SOURCE**

This parameter is optional.

The `EXTERNAL-SOURCE` option can be used in two ways:

1. In order to make a link between the `file-name` definition in MetaSuite and the physical file name on disk. The physical file name will be used automatically in the generated job.

2. In order to indicate that there is no direct access to the data source. The data source might be remote, or there is need to copy or transform the source file via a simple copy or via a transformation tool. In this case the file, specified by the `EXTERNAL-SOURCE` parameter is the file that has to be transferred or copied to the local area.

   The local input file will have the name indicated by `file-name`.

The transfer method is put before the name of the external file, in the form `TTT:FFFFF`, where `TTT` = the name of the transfer method and `FFFFF` = the external file.

Transfer Method Convention.

The treatment of this parameter will be done in the MRL tables.

The client can customize those tables.

Although the interpretation of the `EXTERNAL-SOURCE` option is fully customizable, the standard MetaSuite installation will use the following convention rule:

<table>
<thead>
<tr>
<th>External Source Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP:Filename</td>
<td>The file will be downloaded via the FTP protocol.</td>
</tr>
<tr>
<td>CPY:Filename</td>
<td>A file copy will be done from this file to the standard file.</td>
</tr>
<tr>
<td>STD:Filename</td>
<td>This file name will be taken as input file.</td>
</tr>
</tbody>
</table>

The External Source parameter is not supported yet on every platform.

UNIX and Windows are fully supported. OS390 supports the STD type.

Please contact the MetaSuite support team if you want to use this option on another platform.

Some MTL options are made available in order to help you with the use and customization of your external-source solutions.

Extract from the MTLOPT table:
The MRL tables to be customized are RL1002 and RL1012. RL1002 is used to get the external source on the local machine and RL1012 is used to remove the external source from the local machine at the end of the job. For more information about the use of the MRL tables, please refer to the *Generator Manager User Guide*.

**COLUMN-SEPARATOR**

Optional.

This parameter is used in case of delimited source files.

It contains the delimiter character(s) of the delimited source file.

If this parameter is missing, the default column separator character ";" will be used.

The Column Separator can be 1 to 3 bytes long.

Following special values are allowed:

- SYS-LOW-VALUE or LOW-VALUE : replaces hex "00".
- SYS-HIGH-VALUE or HIGH-VALUE : replaces hex "FF".
- TAB-CHARACTER : replaces hex "09".

**ROW-TERMINATOR**

Optional.

This parameter is used in case of delimited source files.

It contains the character or character sequence on which the record data stops. All bytes behind this character or character sequence are interpreted as comment.

If this parameter is missing, no row terminator character will be interpreted.

The Row Terminator can be 0 to 5 bytes long.

**CCSID**

Optional.

*CCSID* is an abbreviation to mean "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
This **CCSID** can be specified on general MetaSuite level (dictionary settings CHARACTER-CCSID and UNICODE-CCSID), but also on file level, record level and field level.

**XPATH**

Optional.

*XPATH* is a query language for selecting nodes from an XML document. The *XPATH* contains the concatenation of the different nodes that have to be read before the actual content starts.

The *XPATH* delimiter is a forward slash.

Example:

```
ADD FILE COLLECT_TYPES TYPE XML CODE-CONTROL XPC XPATH '/XML/Profiles'
XML-DECLARATION ...
```

No record verification will be done until the nodes `<XML>` and `<Profiles>` have been read. When writing a target file, the full XPATH will be written before the record node is put.

**XML-DECLARATION**

Optional.

This parameter is used when writing XML target files. It contains the XML declaration sentence that must be written at the start of an XML file.

Example:

```
ADD FILE COLLECT_TYPES TYPE XML ... XML-DECLARATION '<?xml version="1.0" encoding="UTF-8"?>'
```

This will result in the following output:

```
<?xml version="1.0" encoding="UTF-8"?>
<xml>
  <Profiles>
    ...
  </Profiles>
<xml>
```

**RULE**

Optional.

The *RULE* option is used to add a business rule documenting your file.

**NOTE**

Optional.

The *NOTE* option is used to add a note documenting your file.

### 25.2. ADD RECORD

The *ADD RECORD* command is used to define a record (table). Each record that is formatted differently should be defined as a separate record. At least one record must be defined for each file.

Note that additional options of the *ADD RECORD* command are available (and in some cases required) when defining the records of database files. Refer to the appropriate database supplement when defining the records of a database file.
ADD RECORD command defines a record in the dictionary file. It requires a record name and can optionally include file name, record size, key field, column separator, row terminator, code set identifier, XML path, and business rule. The record name must follow specific naming conventions and can be referenced in the dictionary file for specific purposes.

### Format

ADD RECORD  
record-name [OF file-name]  
[SIZ[ maximum-record-size]]  
[KEY (field-name value-test,...)]  
[COLUMN-SEPARATOR 'column separator characters']  
[ROW-TERMINATOR 'row terminator characters']  
[CCSID coded-character-set-identifier]  
[XPATH XML-path]  
[RULE business-rule]  
[NOTE note]

### ADD RECORD

**record-name**

Required.  
*Record-name* is the name of the record being defined. The name you enter must meet the following conditions:

- The name may contain up to 32 characters.
- It must begin with an alphabetic character.
- It may contain the characters A-Z and 0-9.
- It may contain the following embedded special characters: $, #, @, _ (underscore) and - (hyphen).

**Note:** Additional restrictions apply to the naming of the records of database files. Refer to the appropriate database supplement when defining such records.

Any appropriate name may be selected for *record-name*. For example, the following command defines a record named EMPLOYEE-DATA within the file named EMPLOYEE-MASTER:

ADD RECORD EMPLOYEE-DATA OF EMPLOYEE-MASTER

**file-name**

Optional.  
*File-name* defines the dictionary file for which the record is defined. When omitted, the last changed dictionary file is considered to be the file for which the record is defined.

**SIZE**

Optional.  
*Maximum-record-size* defines the maximum number of characters included in the record. It must be an integer, and cannot exceed the maximum record size defined for any file in which the record is being defined. If the SIZE option is omitted, it defaults to the record size of the file referenced in the OF list.

**KEY**

KEY (field-name value-test,...)
The **KEY** option is used to identify record identification fields and the specific ranges of values for those key fields. Record identification fields are used to identify the record type being defined. **Field-name** is the name of a field within the record. **Value-test** defines the value of the field that indicates an occurrence of **record-name** has been obtained. **Value-test** takes the forms shown below:

<table>
<thead>
<tr>
<th>Form</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ value</td>
<td>the record-id is equal to the specified</td>
</tr>
<tr>
<td>EQ (value,...)</td>
<td>the record-id is equal to any of the specified values.</td>
</tr>
<tr>
<td>NE value</td>
<td>the record-id is not equal to the specified value.</td>
</tr>
<tr>
<td>NE (value,...)</td>
<td>the record-id is not equal to any of the specified values.</td>
</tr>
<tr>
<td>LT value</td>
<td>the record-id is a value less than the specified value.</td>
</tr>
<tr>
<td>LE value</td>
<td>the record-id is a value that is less than or equal to the specified value.</td>
</tr>
<tr>
<td>GT value</td>
<td>the record-id is a value that is greater than the specified value.</td>
</tr>
<tr>
<td>GE value</td>
<td>the record-id is a value that is greater than or equal to the specified value.</td>
</tr>
<tr>
<td>IR (low TO high)</td>
<td>the record-id is a value that does fall in the specified range of values.</td>
</tr>
<tr>
<td>NI (low TO high)</td>
<td>the record-id is a value that does not fall in the specified range of values.</td>
</tr>
</tbody>
</table>

You can specify any number of expressions, separated by commas. If you do, an occurrence will not be identified unless all the expressions are true.

Assume that a file named TRANSACTION contains three differently formatted records: a master record, a billing record, and a payment record. A one-character field, containing the value "M", "B", or "P", identifies each record type. The three records of this file might be defined as follows:

ADD RECORD MASTER-TRANS KEY (MASTER-TYPE EQ 'M')
ADD RECORD BILL-TRANS KEY (BILL-TYPE EQ 'B')
ADD RECORD PAY-TRANS KEY (PAY-TYPE EQ 'P')

The following example defines two multiple-record types. Each is identified by the value 200 in FIELD-A, but differentiated by the values in FIELD-B. One record, named FIRST-RECORD, is defined by the following command:

ADD RECORD FIRST-RECORD KEY (FIRST-FIELD-A EQ 200, FIRST-FIELD-B EQ 'Z')

The second record is defined with ADD RECORD as shown below:

ADD RECORD SECOND-RECORD KEY (SECOND-FIELD-A EQ 200, SECOND-FIELD-B EQ ('A','B','C'))

Special considerations apply to the coding of the **KEY** option when defining the records of database files. In most cases no **KEY** option is coded for a database record, because only the specific record type requested will be accessed by the DBMS. However, under some circumstances, multiple record types are stored and accessed as a single record type by the DBMS. When this occurs, you must provide a means of distinguishing the different types of records that may be returned. The **KEY** option is used for this purpose, and is described in the appropriate supplement for each supported DBMS.

**COLUMN-SEPARATOR**

Optional.
The `COLUMN-SEPARATOR` option specifies the separator characters to use for a delimited file. This can only be specified for Delimited/BRS file types.

The Column Separator size is one to three characters.

Alphabetic characters are not suitable to serve as column separator. Therefore we have exploited some of them to refer to special characters like TAB, high-value and so forth.

<table>
<thead>
<tr>
<th>Column Separator</th>
<th>Will be replaced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Blank character – space</td>
</tr>
<tr>
<td>L</td>
<td>Low-value character – Hex-00</td>
</tr>
<tr>
<td>H</td>
<td>High-value character – Hex-FF</td>
</tr>
<tr>
<td>T</td>
<td>Tab character – Hex-09</td>
</tr>
</tbody>
</table>

If the user wants to use comma separated value files (CSV-files) as input, then the same abbreviations can be taken. The comma separator for source files can be specified in the MTL option `OPTION-CSV-SEPARATOR`.

**ROW-TERMINATOR**

Optional.

The `ROW-TERMINATOR` option specifies the characters that will end a row of a delimited file. This can only be specified for Delimited file types.

The Row Terminator size is zero to five characters.

**CCSID**

Optional.

`CCSID` is an abbreviation to mean "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.

This `CCSID` can be specified on general MetaSuite level (dictionary settings `CHARACTER-CCSID` and `UNICODE-CCSID`), but also on file level, record level and field level.

**XPATH**

Optional.

`XPATH` is a query language for selecting nodes from an XML document. The `XPATH` contains the concatenation of the different nodes that have to be read before the actual content starts.

The `XPATH` on record level can have an `ABSOLUTE` or a `RELATIVE` value.

- If the value of `XPATH` starts with a forward slash (`"/"`), it is an absolute `XPATH` and the `XPATH` of the file level is being ignored.
- If the value does not start with a forward slash (`"/"`), the value will be concatenated with the `XPATH` on file level. In that case a value must be defined on file level.

See also `XPATH` (page 266) on file level.
RULE
Optional.
The RULE option is used to add a business rule documenting your record.

NOTE
Optional.
The NOTE option is used to add a note documenting your record.

25.3. ADD FIELD
The ADD FIELD command is used to define a field (column). This command defines all of the field's physical attributes, such as size and internal data type. In addition, it may define other (optional) attributes, such as a special editing format, a date format and an allowable range of values.
Note that additional options of the ADD FIELD command are available (and in some cases required) when defining the fields of database files. Refer to the appropriate database supplement when defining the fields of a database file.

Format
ADD FIELD field-name
    [OF { record-name | group-field}]
    POSITION start-position
    SIZE characters
        [OCCURS number-times [DEPENDING ON depend-field]]
        [TYPE datatype ]
        {DATE 'format' | TIME | TIMESTAMP}
        [EDIT 'mask']
        [LIMITS (minimum TO maximum)]
        [NULL-INDICATOR 'Null-indicator']
        [INITIAL initial-value]
        [XML-NAME XML-name]
        [XML-TYPE XML-type]
        [FORMAT-MASK XML-format-mask]
        [CCSID coded-character-set-identifier]
        [XPATH XML-path]
        [RULE business-rule]
        [NOTE note]

ADD FIELD

Field-name

Required.
Field-name is the name of the field being defined.
The name you enter must meet the following conditions:

• The name may contain up to 32 characters. In case of SQL type of fields, 65 characters: 32 characters for the table name, a separating point, and 32 characters for the field name.
• It must begin with an alphabetic character.
• It may contain the characters A-Z and 0-9.
• It may contain the following embedded special characters: $, #, @, _ (underscore) and - (hyphen).
• Field names must be unique in the MetaSuite Generator Library.

As an example, the following commands define two fields, named STATE-CODE and ZIP-CODE:

ADD FIELD STATE-CODE SIZE 2 TYPE CHARACTER
ADD FIELD ZIP-CODE SIZE 5 TYPE ZONED

Record-name

Optional.
Record-name identifies the owning record for the field. If omitted, field-name is assumed to fall in the current record.

Group-field

Optional.
Group-field identifies the encompassing field of a component field. If this specification is omitted, field-name is assumed to fall in the current record.

For example, assume that the first three characters of a nine-digit account number contain a branch number. You might define the account number and branch number sub-field as follows:

ADD FIELD ACCOUNT-NUMBER SIZE 9 TYPE CHARACTER
ADD FIELD BRANCH-CODE OF ACCOUNT-NUMBER SIZE 3 TYPE ZONED UNSIGNED

You may redefine fields with different data types, i.e. character with numeric and numeric with character. Please take the representation of the sign into consideration when you redefine a character as a numeric.

POSITION

Required.
Start-position defines the field’s starting character position in the record or, if the OF group-field option is specified, in the group-field. Start-position must be an integer.

If the record is variable-length, do not include the four-character record descriptor word (at the start of each record) when determining the start position of the field. The first data field starts in position 1.

The following commands illustrate the use of the POSITION option:

ADD FIELD EXP-DATE OF EXPENSE POSITION 21 SIZE 6 TYPE ZONED UNSIGNED DATE 'MMDDYY'
ADD FIELD EXP-YEAR OF EXP-DATE POSITION 5 SIZE 2 TYPE ZONED UNSIGNED

The first field, EXP-DATE, is defined starting in position 21 of the record named EXPENSE. The second field, EXP-YEAR, is defined starting with the fifth character of the field EXP-DATE.

SIZE

Required.
Size specifies the length in bytes of the field being defined. If the field occurs more than one time in the record, the SIZE option defines the length of a single occurrence of the field. Size must be an integer value. Consider the following example:
ADD FIELD CUSTOMER-NAME SIZE 20 TYPE CHARACTER
ADD FIELD CUSTOMER-ACCOUNT SIZE 9 TYPE ZONED

The first command defines the CUSTOMER-NAME field as being 20 characters in length. The second command defines the CUSTOMER-ACCOUNT field as being nine characters in length.

Please refer to the appropriate database supplement for size-specific issues.

If you are using a COBOL record description to define your fields, refer to the following table to determine each field’s size according to the COBOL "use" and "picture clause" definitions. This table also indicates the data type.

<table>
<thead>
<tr>
<th>COBOL Usage</th>
<th>COBOL Picture</th>
<th>MetaSuite Type</th>
<th>MetaSuite Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISPLAY</td>
<td>PIC X(c)</td>
<td>CHARACTER</td>
<td>SIZE c</td>
</tr>
<tr>
<td>PIC X(c)</td>
<td>HEX</td>
<td>SIZE c</td>
<td></td>
</tr>
<tr>
<td>PIC editmask</td>
<td>PRN</td>
<td>SIZE c</td>
<td></td>
</tr>
<tr>
<td>PIC S9(n)V9(d)</td>
<td>ZONED</td>
<td>SIZE n+d+s</td>
<td></td>
</tr>
<tr>
<td>COMPUTATIONAL BINARY</td>
<td>PIC S9(n)V9(d)</td>
<td>BINARY</td>
<td>SIZE 2 1 &lt;= n+d &lt;= 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 4 5 &lt;= n+d &lt;= 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 8 10 &lt;= n+d &lt;= 18</td>
</tr>
<tr>
<td>COMPUTATIONAL-1</td>
<td>FLOAT</td>
<td>SIZE 4</td>
<td></td>
</tr>
<tr>
<td>COMPUTATIONAL-2</td>
<td>FLOAT</td>
<td>SIZE 8</td>
<td></td>
</tr>
<tr>
<td>COMPUTATIONAL-3</td>
<td>PIC S9(n)V9(d)</td>
<td>FLOAT</td>
<td>SIZE (n+d+1)/2</td>
</tr>
<tr>
<td>PACKED-DECIMAL</td>
<td></td>
<td>(rounded up)</td>
<td></td>
</tr>
<tr>
<td>COMPUTATIONAL-5</td>
<td>PIC S9(n)V9(d)</td>
<td>BINARY NATIVE</td>
<td>SIZE 2 1 &lt;= n+d &lt;= 4</td>
</tr>
<tr>
<td></td>
<td>COMP-5</td>
<td></td>
<td>SIZE 4 5 &lt;= n+d &lt;= 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SIZE 8 10 &lt;= n+d &lt;= 18</td>
</tr>
<tr>
<td>COBOL usage NATIONAL</td>
<td>picture N(n)</td>
<td>NATIONAL</td>
<td>2*n</td>
</tr>
<tr>
<td>COBOL usage NATIONAL</td>
<td>picture 9(n)V9(d)</td>
<td>PRN-NATIONAL</td>
<td>2*(n+d)</td>
</tr>
</tbody>
</table>

Where:

- \( c \) = number of characters
- \( n \) = number of integer digits
- \( d \) = number of decimal digits
- \( s \) = 1 for sign indicator (if present) or 0 for no sign indicator

### OCCURS

Optional.

(Number-times) specifies the maximum number of times the field being defined occurs in the record. If the OCCURS option is omitted, the field is assumed to occur one time. When a field occurs more than once, the option is required. Number-times must be an integer. As an example, the command below defines a field that occurs ten times:

ADD FIELD PD-DEDUCTIONS SIZE 3 TYPE PACKED DECIMAL 2 OCCURS 10
**DEPENDING ON**

Optional.

If *field-name* occurs a variable number of times, *number-times* specifies the maximum number of times the field may occur, and *depend-field* specifies the actual number of times the field occurs. The *depend-field* must be a numeric integer field, defined in the same record before *field-name*.

In COBOL, a definition of a variably occurring field might appear as shown below. The key to identifying a variably occurring field from a COBOL definition is the "DEPENDING ON" clause; whenever this clause is present, a variably-occurring field is being defined.

```plaintext
02 ITEM-COUNT    PIC S9(4).
02 ITEM-TABLE    OCCURS 1 TO 100 TIMES DEPENDING ON ITEM-COUNT.
     03 ITEM-NUMBER   PIC 9(5).
     03 ITEM-DESC     PIC X(20).
```

The variably occurring field structure would be defined as shown below:

```plaintext
ADD FIELD ITEM-COUNT SIZE 4 TYPE ZONED
ADD FIELD ITEM-TABLE SIZE 25 OCCURS 100 DEPENDING ON ITEM-COUNT
ADD FIELD ITEM-NUMBER OF ITEM-TABLE POSITION 1 SIZE 5 TYPE ZONED UNSIGNED
ADD FIELD ITEM-DESC OF ITEM-TABLE POSITION 6 SIZE 20 TYPE CHARACTER
```

**TYPE**

Optional.

*Datatype* defines the internal data type of the field. If the option is omitted, TYPE CHARACTER is assumed. A list of possible datatypes is given below:

```plaintext
TYPE {CHARACTER | HEX | BIT number | FLOAT | VARCHAR | BYTE |
     PRN [DECIMAL places] | 
     BINARY [NATIVE] [DECIMAL places] [UNSIGNED] | 
     PACKED [DECIMAL places] [UNSIGNED|NULLSIGN] | 
     ZONED [DECIMAL places] [UNSIGNED||SEPARATE][LEADING]}
```

Each datatype is described separately below.

There are two general classes of datatypes: non-numeric and numeric. Datatypes CHARACTER is considered non-numeric; all others are numeric. (The default for all numeric fields is signed.)
CHARACTER data can be represented by CHARACTER or VARCHAR. TYPE CHARACTER, the default data type, indicates that the field may contain any possible character within the character set being used (including "unprintable" characters). Note that numeric operations cannot be performed on a field defined as CHARACTER, even when that field contains all digits.

Usage: TYPE CHARACTER

Note: Group fields that contain different types of data must be defined as CHARACTER, as illustrated by the example below:

ADD FIELD PRODUCT SIZE 35 TYPE CHARACTER
ADD FIELD PRODUCT-NUMBER OF PRODUCT POSITION 1 SIZE 5 TYPE ZONED UNSIGNED
ADD FIELD PRODUCT-DESCRIPTION OF PRODUCT POSITION 6 SIZE 30 TYPE CHARACTER

The first command defines a 35-character group field named PRODUCT. The first five characters of the PRODUCT field contain a five-digit product number, defined as a ZONED field (meaning that it may contain only the characters 0-9). The last 30 characters are defined as a CHARACTER field, named PRODUCT-DESCRIPTION, which would presumably contain some printable text describing the product.

TYPE VARCHAR gives the character string a variable nature. This data type is useful for supporting relational technology data types.

Usage: TYPE VARCHAR

HEXADECIMAL

TYPE HEX indicates that any hexadecimal characters are allowed within the field being defined.

Usage: TYPE HEX

For example, the following command:

ADD FIELD RECORD-CODE SIZE 1 TYPE HEX

defines a one-character field named RECORD-CODE, which may contain the value X'00' to X'FF'.

BIT

TYPE BIT number indicates that the field being defined occupies a single bit storage.

Usage: TYPE BIT Number

A field defined with the TYPE BIT option may contain only the binary values zero or one. If printed, such a field will print as a character 0 or 1. Number refers to the location of the bit being defined within its encompassing byte, with the highest order (left-most) bit being 1 and the lowest order (right-most) bit being bit 8.

Typically, BIT fields will not be found in files created by COBOL programs, because there is no corresponding COBOL data type. However, you may encounter BIT fields in files created by assembler or other language programs. Numeric operations may be performed on a TYPE BIT field.

The example below illustrates TYPE BIT, and defines the second bit of the third byte in a record:

ADD FIELD CLOSED-FLAG POSITION 3 SIZE 1 TYPE BIT 2

Note that the SIZE specification for a BIT field is always 1, and that the POSITION option refers to the location of the character containing the bit being defined. SQL flags will obtain the data type BIT 1 after collection.
**FLOATING-POINT NUMERIC**

TYPE FLOAT indicates that the field being defined is a floating-point number. A floating-point number is stored in an encoded exponential form. This type of data is rarely used in business applications.

Usage: TYPE FLOAT

Only two SIZE specifications are allowed for FLOAT fields: 4 or 8 (characters in length), allowing, respectively, for 8 or 17 digits of precision.

The following commands illustrate the two types of floating-point numeric fields that may be defined:

ADD FIELD ESTIMATED-GRAINS-OF-SAND SIZE 4 TYPE FLOAT
ADD FIELD MORE-PRECISE-ESTIMATE SIZE 8 TYPE FLOAT

These two types of numeric data are sometimes referred to as "short" and "long" floating-point numeric (or as "single precision" and "double precision" numeric, respectively).

**PRINTED NUMERIC**

TYPE PRN indicates that the field contains a printed numeric value. The format in which the printed numeric value is displayed is to be specified by the mandatory option 'EDIT'. The PRN value needs to be converted before it can be used.

MetaSuite Generator will automatically convert this value when required.

Usage: TYPE PRN [DECIMAL Places] EDIT 'mask'

The following commands illustrate the definition of two PRN fields. The first field has no decimal place, the second field has two decimal places.

ADD FIELD employee_number POSITION 1 SIZE 8 TYPE PRN EDIT '-9999999' NULL-INDICATOR 'NOTNULL'
ADD FIELD annual_salary POSITION 1 SIZE 11 TYPE PRN DECIMAL 2 EDIT '-9999999.99' NULL-INDICATOR 'OUTNULL'

**BINARY**

TYPE BINARY indicates that the field contains a binary numeric value: one that is stored as a sequence of 0's and 1's. Only three sizes are allowed for binary fields: half-word (two bytes), full-word (four bytes) and double-word (eight characters).

Usage: TYPE BINARY [DECIMAL Places] [UNSIGNED]

The DECIMAL option indicates the number of decimal places to the right of an implied decimal point. Omit the DECIMAL option for integers; it defaults to 0 places.

The UNSIGNED option indicates that there is no sign present on the data.

The following commands illustrate the definition of three binary numeric fields, each of a different size. The first two fields have no decimal places, and the third field has two decimal places.

ADD FIELD FLIGHT-TYPE SIZE 2 TYPE BINARY
ADD FIELD FLIGHT-NUMBER SIZE 4 TYPE BINARY
ADD FIELD CUM-FLIGHT-MILES SIZE 8 TYPE BINARY DECIMAL 2

**BINARY NATIVE**

TYPE BINARY NATIVE indicates that the field contains a binary numeric value: one that is stored as a sequence of 0's and 1's. Only three sizes are allowed for binary fields: half-word (two bytes), full-word (four bytes) and double-word (eight characters). Internal storage depends on the operating system.

Usage: TYPE BINARY NATIVE [DECIMAL Places] [UNSIGNED]

The DECIMAL option indicates the number of decimal places to the right of an implied decimal point. Omit the DECIMAL option for integers; it defaults to 0 places. The UNSIGNED option indicates that there is no sign present on the data.

The following commands illustrate the definition of three binary numeric fields, each of a different size. The first two fields have no decimal places, and the third field has two decimal places.

ADD FIELD FLIGHT-TYPE SIZE 2 TYPE BINARY NATIVE
ADD FIELD FLIGHT-NUMBER SIZE 4 TYPE BINARY NATIVE
ADD FIELD CUM-FLIGHT-MILES SIZE 8 TYPE BINARY NATIVE DECIMAL 2
PACKED

TYPE PACKED indicates that the field contains a packed decimal value.
Usage: TYPE PACKED [DECIMAL Places] [UNSIGNED | NULLSIGN]
The DECIMAL option indicates the number of decimal places to the right of an
implied decimal point. Omit the DECIMAL option for integers; it defaults to 0
places.
Packed decimal is the most commonly used internal numeric data type. Two decimal
digits are contained in each byte of a packed decimal number, except that, if the
number is signed, the last half byte contains a positive or negative sign indicator
(and no digit). Thus, to calculate the size of a packed decimal numeric, you should
use the following formula:

\[ \text{SIZE} = \frac{n + s}{2} \]

Rounded up to the next integer, the value of n is the number of digits in the number.
s is 1 if the value is signed; 0 if it is not.
The following commands illustrate the definition of two packed decimal fields:
ADD FIELD SALES-QUARTER SIZE 1 TYPE PACKED
ADD FIELD SALES-AMOUNT SIZE 8 TYPE PACKED DECIMAL 2
Because it is only one character in length, the SALES-QUARTER field may contain
only the value -9 through +9. The second field, SALES-AMOUNT, may contain up to
15 digits (including two decimal places), in addition to the sign indicator. It is
important to note that any packed decimal field created by most COBOL programs
will always contain a sign indicator in the last half byte. Packed decimal numeric
fields created by other language programs may or may not contain the sign
indicator. If there is no sign indicator, the UNSIGNED option must be coded.
Whenever you redefine the leading digits of a packed number as a separate field,
you will specify the UNSIGNED option, as illustrated by the example below:
ADD FIELD SHP-DATE SIZE 3 TYPE PACKED DATE 'YYDDD'
ADD FIELD SHP-YEAR OF SHP-DATE POSITION 1 SIZE 1 TYPE PACKED UN-
SIGNED
SHP-YEAR redefines the first two digits of the packed field SHP-DATE.
If a field is defined a packed decimal field, and within a generated program it is
found to contain invalid decimal digits or an invalid sign indicator, an error message
("INVALID NUMERIC DATA") will be produced by the system.
The NULLSIGN option defines a packed field in which the second half of the last
byte has a neutral sign (value X'xF').

NATIONAL

A national character string literal is a literal having 2-byte Unicode characters as its
value.
The most common characters of the local character set (CCSID) can be derived from
the ASCII character set and a low-value added before (big endian systems) or behind
(little endian systems).

PRN-NATIONAL

This is the NATIONAL variant of PRN.
Note: The same restrictions as for PRN apply to PRN-NATIONAL. The size of the
field will be twice the size of the mask.
ZONED

**TYPE ZONED** indicates that the field contains decimal numbers in a printable character format, with one digit stored per byte.

**Usage:** TYPE ZONED [DECIMAL Places] [UNSIGNED] [SEPARATE] [LEADING]

The **DECIMAL** option indicates the number of decimal places to the right of an implied decimal point. Omit the **DECIMAL** option for integers; it defaults to 0 places.

**TYPE ZONED** numbers are sometimes referred to as "display" or "external" decimal numbers. Leading blanks are not permitted in this type of field, nor may it contain editing characters, such as commas or a decimal-point (which are permitted in COBOL edited numeric or print numeric fields, and are represented by MetaSuite datatype PRN).

The following commands illustrate the definition of two zoned decimal numbers.
The first field contains two digits and the second field contains three digits.

```
ADD FIELD DIVISION-CODE SIZE 2 TYPE ZONED
ADD FIELD NEW-BRANCHES SIZE 3 TYPE ZONED
```

A sign indicator is typically "over-punched" on the last character of a zoned numeric. If there is no "over-punched" sign in the number, you must specify the **UNSIGNED** option. Whenever you redefine the leading digits of a zoned number as a separate field, you will specify the **UNSIGNED** option, as illustrated by the example below (where DUE-MONTH redefines the first two digits of the zoned numeric field DUE-DATE):

```
ADD FIELD DUE-DATE SIZE 6 TYPE ZONED DATE 'MMDDYY'
ADD FIELD DUE-MONTH OF DUE-DATE POSITION 1 SIZE 2 TYPE ZONED UNSIGNED
```

There are three options, as follows:

- **LEADING** puts the sign on the initial digit.
- **SEPARATE** puts the sign in the last byte (no digit).
- **SEPARATE LEADING** puts the sign in the initial byte (no digit).

If the **ZONED** number contains a separate (i.e., not over-punched) plus or minus sign attached to the number, you must specify the **SEPARATE** option. In the first example above, if the second field (NEW-BRANCHES) contained a separate plus or minus sign, it would have been defined with the **SEPARATE** option, and could contain only two digits (since one character position would be occupied by the sign).

When a separate sign is included in a field, it is generally at the end of the numeric position of the field. If, instead, the sign appears at the start of the number, then you must include the keyword **LEADING**.

The example below defines a three-digit zoned numeric field that is followed by a plus or minus sign. If the sign preceded the digits, the **LEADING** option would have been included.

```
ADD FIELD DAYS-LATE SIZE 4 TYPE ZONED SEPARATE
```

The **SIZE** specification for a zoned numeric is equal to the number of digits. An exception occurs when the **SEPARATE** option is coded, in which case the size is equal to the number of digits plus one character (for the sign).

If you are using a COBOL record description to define your fields, refer to the table in the section **SIZE** (page 271) to determine each field’s data type according to the COBOL "use" and "picture clause" definitions.

---

**DATE**

Optional.

The **DATE** option defines the field as a date field. The system automatically validates date fields whenever they are referenced in program requests, and automatically converts date fields whenever they are compared to another date or used in a calculation.
**Format** is specified as a string of Ys, Ms and Ds, representing, respectively, digits for years, months and days. A ? indicates any single character like -, / and . . The format specification must be enclosed in single quotation marks.

Allowable format values are listed below:

\[
'MMDDYY' \quad 'DDMMYY' \quad 'MMDDYYYY' \quad 'DDMMYYYY' \\
'YYMMDD' \quad 'YYDDMM' \quad 'YYYYMMDD' \quad 'YYYYDDMM' \\
'YYDDD' \quad 'YYYYDDD' \quad 'YY?DDD' \quad 'YYYY?DDD'
\]

The format specification is used by the system to choose the appropriate conversion routine, when validating dates or performing calculations or comparisons based on dates.

For example, if the following two fields were defined:

```
ADD FIELD PAYMENT-DATE SIZE 6 TYPE ZONED DATE 'DDMMYY'
ADD FIELD SALE-DATE SIZE 4 TYPE PACKED DATE 'YYMMDD'
```

the system would always check to determine that PAYMENT-DATE contained a valid date in the DDMMYYYY format, and that SALE-DATE contained a valid date in the YYMMDD format. For example, if PAYMENT-DATE contained the value 010195, and SALE-DATE had the value 940805, the following conditional command:

```
IF PAYMENT-DATE GT SALE-DATE...
```

This statement would test as "true" with no conversions having to be performed by the user in procedural code.

**TIME**

Optional.

The **TIME** option is used to define a field of type **TIME**. The field can be any TYPE and SIZE. For example, you might define:

```
ADD FIELD HHMMSS TYPE ZONED SIZE 6 TIME
```

**TIMESTAMP**

Optional.

The **TIMESTAMP** option is used to define a field of type **TIMESTAMP**. For example:

```
ADD FIELD SET-TIMESTAMP TYPE CHARACTER SIZE 26 TIMESTAMP
```

**EDIT**

Optional.

The **EDIT** option is used to override the default (output) edit mask for the field. It allows you to define the output format *(mask)* for the field, specifying what types of characters can be printed, and insertion characters -- such as blanks and commas -- that will improve the readability of the output reports.

You can also use the edit mask of a field on a non-report, when the SYS-EDIT function is used. An edit mask consists of a combination of symbols, enclosed in quotation marks, that describe the contents of the field and allow the insertion of special editing characters. When constructing an edit mask, you specify what type of character appears in each position of the printed field. To do this, you place a symbol in the corresponding position of the mask.

Repeating editing characters can be noted as '?(n)' where ? = the repeating character and n = the number of occurrences.
There are two types of editing characters, called "replacement" and "insertion" characters. These types of characters are described below.

REPLACEMENT Characters

Replacement characters indicate positions in the printed field that may be replaced by (the corresponding types of) characters from the input field. Possible replacement characters are listed below:

<table>
<thead>
<tr>
<th>Replacement Character</th>
<th>Description of Replaced Character(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Floating dollar sign before the first digit, with leading zero suppression</td>
</tr>
<tr>
<td>Z</td>
<td>Leading zero suppression</td>
</tr>
<tr>
<td>N</td>
<td>2-byte based Unicode</td>
</tr>
<tr>
<td>*</td>
<td>Asterisks to replace leading zeros</td>
</tr>
<tr>
<td>9</td>
<td>Numeric character</td>
</tr>
<tr>
<td>A</td>
<td>Alphabetic character</td>
</tr>
<tr>
<td>X</td>
<td>Alphanumeric character</td>
</tr>
</tbody>
</table>

INSERTION Characters

Insertion characters indicate characters to be printed in addition to those contained in the stored field. Possible insertion characters are listed below:

<table>
<thead>
<tr>
<th>Insertion Character</th>
<th>Description of Inserted Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Leading dollar sign</td>
</tr>
<tr>
<td>*</td>
<td>Leading asterisk (generally for check protection)</td>
</tr>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>.</td>
<td>Decimal Point</td>
</tr>
<tr>
<td>B</td>
<td>Blank</td>
</tr>
<tr>
<td>-</td>
<td>Trailing minus sign for negative values</td>
</tr>
<tr>
<td>+</td>
<td>Trailing plus or minus sign</td>
</tr>
<tr>
<td>CR</td>
<td>Trailing credit symbol for negative values only</td>
</tr>
<tr>
<td>DB</td>
<td>Trailing debit symbol for negative values</td>
</tr>
</tbody>
</table>

By default, the system provides you with the most commonly used edit mask for a field, depending on its data type and other definition options specified.
The components of the default edit mask are determined as follows:

- For **signed numeric fields**, the default edit mask will contain a minus sign as the right-most character. All negative values will print with a trailing minus sign.
- For **non-integral numeric fields**, the default edit mask will contain a decimal point and as many digit replacement characters (9s) to the right of the decimal point as are specified by the DECIMAL option.
- For **numeric fields**, the default edit mask will contain comma insertion characters as appropriate for any value greater than 999, and zero suppression characters (Zs) for each digit to the left of the decimal point (if any). If no DECIMAL option has been coded the rightmost digit will be represented by a digit replacement character (9), rather than a zero suppression character (Z).
- For **date fields**, a special routine is used to insert slashes between the month, day, and year values. The default edit mask for a field defined with the DATE option may not be overridden.
- For **alphanumeric fields**, the default edit mask contains as many alphanumeric character replacement characters (Xs) as are required to print the field.

The table below illustrates the generated system default edit masks for various types of fields:

<table>
<thead>
<tr>
<th>Field Definition</th>
<th>Default Edit Mask</th>
<th>Field Value</th>
<th>Printed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE 6 TYPE CHARACTER</td>
<td>‘X(6)’</td>
<td>AB138C</td>
<td>AB138C</td>
</tr>
<tr>
<td>SIZE 2 TYPE ZONED DECIMAL 2</td>
<td>‘.99-’</td>
<td>350</td>
<td>.35 .00</td>
</tr>
<tr>
<td>SIZE 4 TYPE ZONED</td>
<td>‘Z,ZZ9.-’</td>
<td>231</td>
<td>231</td>
</tr>
<tr>
<td>SIZE 4 TYPE ZONED</td>
<td>‘Z,ZZ9.-’</td>
<td>-231</td>
<td>231-8,234</td>
</tr>
<tr>
<td>SIZE 5 TYPE ZONED DECIMAL 2</td>
<td>‘ZZ9.99-’</td>
<td>12.88</td>
<td>12.88</td>
</tr>
</tbody>
</table>

To avoid interpunction in the default edit masks, use the **CHANGE DEFAULT INTERPUNCATION** command (MIL) in the Generator Manager.

This command is used to specify the default edit mask for numeric values (if the CODE flag is not set). If the CODE flag is set, the default edit mask will have no zero suppression and no interpunction, regardless of the INTERPUNCATION option.

For more information, refer to the chapter *The Generate Screen - Implementing MIL Instructions* in the *Generator Manager User Guide*.

When overriding the system default edit mask for a field, you must account for all of the characters in the field (unless it is your intention to truncate the field on output). The table below illustrates the output produced by common edit mask specifications:
<table>
<thead>
<tr>
<th>Edit Specification</th>
<th>Field Value</th>
<th>Printed Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDIT 'XXBXXXX'</td>
<td>AB138C</td>
<td>AB138C</td>
</tr>
<tr>
<td>EDIT '.99'</td>
<td>.35</td>
<td>.35</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>.00</td>
</tr>
<tr>
<td></td>
<td>-.12</td>
<td>.12</td>
</tr>
<tr>
<td>EDIT 'ZZZZ'</td>
<td>231</td>
<td>231</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>(blank)</td>
</tr>
</tbody>
</table>

Null-Indicator (former DBNAME)

Optional.

The Null-Indicator is used to specify the nullability of a field, and whether Inbound or Outbound nulls are used for the field.

NULL-INDICATOR ' (NOTNULL | INNULL | OUTNULL | OUTNULR | DEFAULT) '

Each type of Null-indicator is described separately.

Note: If you do not include a sign indicator character for numeric fields, negative values will be printed as positive values in your reports, as illustrated in the table above.
### NOTNULL
The field is a NOTNULL field, and no NULL value should be assigned in the MetaMap model to this field. You should always assign a value to a NOTNULL field. For example:

```sql
ADD FIELD department OF PP_department POSITION 1 SIZE 15 TYPE CHARACTER NULL-INDICATOR 'NOTNULL'
```

### INNULL
The field is a NULLS Allowed field, and a NULL value can be assigned in the MetaMap model to this field. When a NULL value is assigned, the first position of the field itself indicates the NULL value (the so called inbound NULL). For example:

```sql
ADD FIELD employee_count OF PP_department POSITION 16 SIZE 2 TYPE BINARY NULL-INDICATOR 'INNULL'
```

The value of position 16 determines whether employee_count is NULL, or whether it contains a real value.

### OUTNULL
The field is a NULLS Allowed field, and a NULL value can be assigned in the MetaMap model to this field. To store an eventual NULL value an additional byte is foreseen in the sequential file that precedes the real field. When a NULL value is assigned, this additional position indicates the NULL value (the so called outbound NULL). For example:

```sql
ADD FIELD dept_annual_salary OF PP_department POSITION 18 SIZE 5 TYPE PACKED DECIMAL 2 NULL-INDICATOR 'OUTNULL'
```

The value of position 17 determines whether dept_annual_salary is NULL, or whether the value that is stored from position 18 onwards is the real value of the field.

### OUTNULR
The field is a NULLS Allowed field, and a NULL value can be assigned in the MetaMap model to this field. To store an eventual NULL value an additional byte is foreseen in the sequential file that follows the real field. When a NULL value is assigned, this additional position indicates the NULL value (the so called outbound NULL). For example:

```sql
ADD FIELD dept_annual_salary OF PP_department POSITION 18 SIZE 5 TYPE PACKED DECIMAL 2 NULL-INDICATOR 'OUTNULR'
```

The value of position 23 determines whether dept_annual_salary is NULL, or whether the value that is stored from position 18 onwards is the real value of the field.

### DEFAULT
The field is a NOTNULL field, and no NULL value should be assigned in the MetaMap model to this field. When no value is assigned to a DEFAULT NOTNULL field, some database systems provide you with the possibility to assign a default value to the field. For example:

```sql
ADD FIELD department OF PP_department POSITION 1 SIZE 15 TYPE CHARACTER NULL-INDICATOR 'DEFAULT'
```

**Note:** The use of Null-Indicator can change when the field describes a database field. Please refer to the appropriate database supplement for instruction. If the NULL-INDICATOR parameter is left out, the standard "NULLABLE" value will be taken. This value can be either "INNULL" or "NOTNULL". Please refer to the Generator Manager User Guide for more information on the "NULLABLE" parameter.
INITIAL
Optional.
The INITIAL option is used in case of target file MDL definitions.
In case of source file definitions this parameter will not affect any operation, except for syntax checking.
MetaStore Manager can collect MDL definitions containing the INITIAL option.
SYS-LOW-VALUE and SYS-HIGH-VALUE can also be used.
In case of target files, MetaMap will not write the original MDL definition to the MXL file. For target files MetaMap will convert the ADD FIELD commands into FIELD commands. Please refer to the FIELD command in the MetaMap Manager User Guide for more information on this subject.

XML-NAME
Optional.
The collected name of the XML node. In most cases, the XML-name will be a part of the field name.

XML-TYPE
Optional.
This XML-TYPE can be GROUP, NODE or ATTRIBUTE. NODE is the default value.
This property is important for the correct output of the XML file, when this definition is used in a target file.
In case of reading XML files, this property is not important.

FORMAT-MASK
Optional.
This field can contain the XML edit mask. (MetaStore Manager does it automatically during the XSD collect process). It will be used in later versions.

CCSID
Optional.
CCSID is an abbreviation to mean "Coded Character Set Identifier". It is a 16-bit number that represents a specific encoding of a specific code page.
This CCSID can be specified on general MetaSuite level (dictionary settings CHARACTER-CCSID and UNICODE-CCSID), but also on file level, record level and field level.

XPATH
Optional.
XPATH, the XML Path, is a format for selecting nodes from an XML document.
The XPATH contains the different nodes that have to be read before the real "file" content starts. The XPATH on file level -if specified- should start with a forward slash (/).

RULE
Optional.
The RULE option is used to add a business rule documenting your field.
NOTE
Optional.
The NOTE option is used to add a note documenting your field.

25.4. COPY FILE
The COPY FILE command is used to copy one or more MetaSuite Generator dictionary files to an output command file. The file to contain the copied commands is TEMPDIR\MSCOPY.MDL.
Where TEMPDIR is the directory defined in the TEMP system/user variable, the TMP directory defined in the TMP system/user variable, or the directory x:\installationdirectory\GENxxx\TMP.

Format
COPY FILE [ file-name | ALL ]
File-name is the name of the MetaSuite Generator dictionary file to be copied to the output command file. For example, to copy the definition of the dictionary file EMPLOYEE-MASTER to the output command file, you would use the following command:

COPY FILE EMPLOYEE-MASTER
ALL indicates that all the dictionary files defined in the MetaSuite Generator library need to be copied to the output command file.

25.5. COPY RECORD
The COPY RECORD command is used to copy one or more MetaSuite Generator dictionary records to an output command file. The file to contain the copied commands is TEMPDIR\MSCOPY.MDL.
Where TEMPDIR is the directory defined in the TEMP system/user variable, the TMP directory defined in the TMP system/user variable, or the directory x:\installationdirectory\GENxxx\TMP.

Format
COPY RECORD [ Record-name | ALL OF FILE file-name | ALL]
Record-name is the name of the MetaSuite Generator dictionary record to be copied to the output command file. For example, to copy the definition of the dictionary record EMPLOYEE-DATA to the output command file, you would use the following command:

COPY RECORD EMPLOYEE-DATA
File-name is the name of the MetaSuite Generator dictionary file for which all its dictionary records are to be copied to the output command file. For example, to copy the definition of all the dictionary records of the dictionary file EMPLOYEE-MASTER to the output command file, you would use the following command:

COPY RECORD ALL OF FILE EMPLOYEE-MASTER
ALL indicates that all the dictionary records defined in the MetaSuite Generator library need to be copied to the output command file.
25.6. COPY FIELD

The **COPY FIELD** command is used to copy one or more MetaSuite Generator dictionary fields to an output command file. The file to contain the copied commands is TEMPDIR\MSCOPY.MDL.
Where TEMPDIR is the directory defined in the TEMP system/user variable, the TMP directory defined in the TMP system/user variable, or the directory x:installationdirectory\GENxxx\TMP.

**Format**

COPY FIELD [ Field-name | ALL OF RECORD Record-name | ALL OF FILE file-name | ALL]

*Field-name* is the name of the MetaSuite Generator dictionary field to be copied to the output command file. For example, to copy the definition of the dictionary field EMPLOYEE-NUMBER to the output command file, you would use the following command:

COPY FIELD EMPLOYEE-NUMBER

*Record-name* is the name of the MetaSuite Generator dictionary record for which all its dictionary fields are to be copied to the output command file. For example, to copy the definition of all the dictionary fields of the dictionary record EMPLOYEE-DATA to the output command file, you would use the following command:

COPY FIELD ALL OF RECORD EMPLOYEE-DATA

*File-name* is the name of the MetaSuite Generator dictionary file for which all its dictionary fields are to be copied to the output command file. For example, to copy the definition of all the dictionary fields of the dictionary file EMPLOYEE-MASTER to the output command file, you would use the following command:

COPY FIELD ALL OF FILE EMPLOYEE-MASTER

*ALL* indicates that all the dictionary fields defined in the MetaSuite Generator library need to be copied to the output command file.

25.7. DELETE FILE

The **DELETE FILE** command is used to delete a dictionary file from the MetaSuite Generator library

**Format**

DELETE FILE file-name

*File-name* is the name of the MetaSuite Generator dictionary file to be deleted. For example, to delete the dictionary file named EMPLOYEE-MASTER from the library, you might use the following command:

DELETE FILE EMPLOYEE-MASTER

25.8. LIST FILE

The **LIST FILE** command is used to list a dictionary file in the MetaSuite Generator library. This command is particularly useful to produce a hard copy listing of each MetaSuite Generator library dictionary file before modifying or deleting it.
Format

LIST FILE [ file-name | ALL ]

_file-name_ is the name of the MetaSuite Generator dictionary file to be listed. For example, to list the
definition of the dictionary file EMPLOYEE-MASTER, you would use the following command:

LIST FILE EMPLOYEE-MASTER

_ALL_ indicates that all the dictionary files defined in the MetaSuite Generator library need to be listed.

25.9. LIST RECORD

The _LIST RECORD_ command is used to list a dictionary record in the MetaSuite Generator library. This
command is particularly useful to produce a hard copy listing of each MetaSuite Generator library dictionary
record before modifying or deleting it.

Format

LIST RECORD [ Record-name | ALL OF FILE file-name | ALL]

_Record-name_ is the name of the MetaSuite Generator dictionary record to be listed. For example, to list the
definition of the dictionary record EMPLOYEE-DATA, you would use the following command:

LIST RECORD EMPLOYEE-DATA

_File-name_ is the name of the MetaSuite Generator dictionary file for which all its dictionary records are to be
listed. For example, to list the definition of all the dictionary records of the dictionary file EMPLOYEE-
MASTER, you would use the following command:

LIST RECORD ALL OF FILE EMPLOYEE-MASTER

_ALL_ indicates that all the dictionary records defined in the MetaSuite Generator library need to be listed.

25.10. LIST FIELD

The _LIST FIELD_ command is used to list a dictionary field in the MetaSuite Generator library. This
command is particularly useful to produce a hard copy listing of each MetaSuite Generator library dictionary
field before modifying or deleting it.

Format

LIST FIELD [ Field-name | ALL OF RECORD Record-name | ALL OF FILE file-name | ALL]

_Field-name_ is the name of the MetaSuite Generator dictionary field to be listed. For example, to list the
definition of the dictionary field EMPLOYEE-NUMBER, you would use the following command:

LIST FIELD EMPLOYEE-NUMBER

_Record-name_ is the name of the MetaSuite Generator dictionary record for which all its dictionary fields are to
be listed. For example, to list the definition of all the dictionary fields of the dictionary record EMPLOYEE-
DATA, you would use the following command:
LIST FIELD ALL OF RECORD EMPLOYEE-DATA

*File-name* is the name of the MetaSuite Generator dictionary file for which all its dictionary fields are to be listed. For example, to list the definition of all the dictionary fields of the dictionary file EMPLOYEE-MASTER, you would use the following command:

LIST FIELD ALL OF FILE EMPLOYEE-MASTER

*ALL* indicates that all the dictionary fields defined in the MetaSuite Generator library need to be listed.
## Index

### A

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adabas File Group</td>
<td>21</td>
</tr>
<tr>
<td>Creating the Dictionary File</td>
<td>21</td>
</tr>
<tr>
<td>Defining Fields</td>
<td>28</td>
</tr>
<tr>
<td>Defining Indexes</td>
<td>42</td>
</tr>
<tr>
<td>Defining Records</td>
<td>24</td>
</tr>
<tr>
<td>Defining Relationships</td>
<td>38</td>
</tr>
<tr>
<td>ADD FIELD</td>
<td>270</td>
</tr>
<tr>
<td>ADD FILE</td>
<td>258</td>
</tr>
<tr>
<td>ADD RECORD</td>
<td>266</td>
</tr>
</tbody>
</table>

### B

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch</td>
<td>254</td>
</tr>
<tr>
<td>Building Dictionary Files Manually</td>
<td></td>
</tr>
<tr>
<td>Adabas File Groups</td>
<td>21</td>
</tr>
<tr>
<td>Datacom File Groups</td>
<td>46</td>
</tr>
<tr>
<td>IDMS Subschemas</td>
<td>134</td>
</tr>
<tr>
<td>IMS PCB</td>
<td>71</td>
</tr>
<tr>
<td>SQL Table Groups</td>
<td>92</td>
</tr>
<tr>
<td>Standard Files</td>
<td>109</td>
</tr>
<tr>
<td>Supra Database</td>
<td>159</td>
</tr>
</tbody>
</table>

### C

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collecting Dictionary Files</td>
<td>180</td>
</tr>
<tr>
<td>Collect File Screen</td>
<td>181</td>
</tr>
<tr>
<td>Collecting Sources</td>
<td></td>
</tr>
<tr>
<td>IDMS</td>
<td>188</td>
</tr>
<tr>
<td>IMS</td>
<td>193</td>
</tr>
<tr>
<td>RDBMS</td>
<td>194</td>
</tr>
<tr>
<td>Sequential</td>
<td>183</td>
</tr>
<tr>
<td>Unload Sequential</td>
<td>198</td>
</tr>
<tr>
<td>XML Schema</td>
<td>211</td>
</tr>
<tr>
<td>Collecting Targets</td>
<td>212</td>
</tr>
<tr>
<td>Load Delimited</td>
<td>223</td>
</tr>
<tr>
<td>Load Sequential</td>
<td>212</td>
</tr>
<tr>
<td>RDBMS</td>
<td>235</td>
</tr>
<tr>
<td>Commands</td>
<td>258</td>
</tr>
<tr>
<td>ADD FIELD</td>
<td>270</td>
</tr>
<tr>
<td>ADD FILE</td>
<td>258</td>
</tr>
<tr>
<td>ADD RECORD</td>
<td>266</td>
</tr>
<tr>
<td>COPY FIELD</td>
<td>285</td>
</tr>
<tr>
<td>COPY FILE</td>
<td>284</td>
</tr>
<tr>
<td>COPY RECORD</td>
<td>284</td>
</tr>
<tr>
<td>DELETE FILE</td>
<td>285</td>
</tr>
</tbody>
</table>

### D

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datacom File Group</td>
<td>46</td>
</tr>
<tr>
<td>Creating the Dictionary File</td>
<td>46</td>
</tr>
<tr>
<td>Defining Fields</td>
<td>52</td>
</tr>
<tr>
<td>Defining Indexes</td>
<td>49</td>
</tr>
<tr>
<td>Defining Records</td>
<td>137</td>
</tr>
<tr>
<td>Defining Relationships</td>
<td>63</td>
</tr>
<tr>
<td>Definition Language Commands</td>
<td>258</td>
</tr>
<tr>
<td>DELETE FIELD</td>
<td>285</td>
</tr>
<tr>
<td>Dictionary File Creation</td>
<td></td>
</tr>
<tr>
<td>Collecting</td>
<td>180</td>
</tr>
<tr>
<td>Importing MDL Files</td>
<td>241</td>
</tr>
<tr>
<td>Manually</td>
<td>20</td>
</tr>
<tr>
<td>Overview</td>
<td>19</td>
</tr>
</tbody>
</table>

### E

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exporting Dictionary Files</td>
<td>242</td>
</tr>
</tbody>
</table>

### I

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDMS Subschemas</td>
<td>134</td>
</tr>
<tr>
<td>Creating the Dictionary File</td>
<td>135</td>
</tr>
<tr>
<td>Defining Fields</td>
<td>140</td>
</tr>
<tr>
<td>Defining Indexes</td>
<td>155</td>
</tr>
<tr>
<td>Defining Records</td>
<td>137</td>
</tr>
<tr>
<td>Defining Relationships</td>
<td>151</td>
</tr>
<tr>
<td>Importing MDL Files</td>
<td>241</td>
</tr>
<tr>
<td>IMS PCB</td>
<td></td>
</tr>
<tr>
<td>Creating the Dictionary File</td>
<td>71</td>
</tr>
<tr>
<td>Defining Fields</td>
<td>77</td>
</tr>
<tr>
<td>Defining Indexes</td>
<td>87</td>
</tr>
<tr>
<td>Defining Records</td>
<td>74</td>
</tr>
<tr>
<td>Internal Source Control</td>
<td>250</td>
</tr>
</tbody>
</table>
L
LIST FIELD 286
LIST FILE 285
LIST RECORD 286

M
MetaStore Manager
Getting Started 6
Key notions 4
Prerequisites 5
Purpose 3

S
Sources
IDMS 188
IMS 193
RDBMS 194
Sequential 183
Unload Sequential 198
XML Schema 211
SQL Table Group 92
Creating the Dictionary File 92
Defining Fields 99
Defining Records 97
Standard File 109
Creating the Dictionary File 109
Defining Fields 122
Defining Records 117
Supra Database 159
Creating the Dictionary File 159
Defining Fields 164
Defining Records 161
Defining Relationships 175

T
Targets
Load Delimited 223
Load Sequential 212
RDBMS 235

U
User Interface
Docking a Window 17
Menu Bar 8
Output Window 15
Record Fields Window 13
Record Layout Window 16
Statusbar 17
Toolbar 10
Tree Window 11
Workspace 12

User Profiles 244
Version Management with Source Control 245