EasternGraphics

visualize your business

OFML–Datenstruktur und -registrierung (DSR)

Spezifikation Version 3.6

Editoren: Bernd Heinemann, Stefan Bleuel, Thomas Gerth

1. August 2023

Rechtliche Hinweise

Inhaltsverzeichnis

Literatur

[asv] Artiklespezifische Ansichten (ASV) Spezifikation. EasternGraphics GmbH

[gln] GLN/ILN http://de.wikipedia.org/wiki/Global_Location_Number

[glos] Bibliotheken, Serien & Co. – Wesentliche OFML–Begriffe. EasternGraphics GmbH

[gtin] GTIN http://de.wikipedia.org/wiki/Global_Trade_Item_Number

[iso639-1] ISO 639-1 http://de.wikipedia.org/wiki/ISO_639

[iso3166] ISO 3166 http://de.wikipedia.org/wiki/ISO_3166

[iso8601] ISO 3166 http://de.wikipedia.org/wiki/ISO_8601

[oap] OFML Aided Planning (OAP) Spezifikation. EasternGraphics GmbH

[ofml] OFML – Standardisiertes Datenbeschreibungsformat der Büromöbelindustrie. Industrieverband Büro und Arbeitswelt e. V. (IBA)

[ppr] Preisprofile (PPR) in Herstellerdaten zur Verarbeitung mit pCon-Produkten. EasternGraphics GmbH

[utc] UTC http://de.wikipedia.org/wiki/Koordinierte_Weltzeit

[xcf] Extensible Catalog Format (XCF) Spezifikation. EasternGraphics GmbH

1 Vorbemerkungen

Bei Dateinamen ist unbedingt darauf zu achten, dass die Klein-/Großschreibweise genau der Bildungsvorschrift entspricht. Der Dateiname muss exakt so geschrieben werden, wie die Schlüssel in der Registrierungsdatei hinterlegt sind. Diese Konvention ist zwingend, um die Lauffähigkeit der OFML–Daten (s. [ofml]) plattformunabhängig zu gewährleisten. Prinzipiell ist die Kleinschreibweise zu bevorzugen.

Für alle Pfadangaben wird in der Spezifikation das Slash–Zeichen ('/') verwendet. Alternativ kann auch der Backslash '\' verwendet werden.

Die in dieser Spezifikation definierten Schlüssel und Schlüsselwerte müssen in exakt der hier vorgegebenen Schreibweise verwendet werden.

Folgende Stile werden zur Kennzeichnung der verschiedenen Elemente verwendet:

`Beispiel`

`Datei- bzw. Verzeichnisangabe`

Schlüsseldefinition

`Schlüsselreferenz`

Die Kennung hinter dem Schlüsselnamen gibt an:

[M]	unbedingter Pflichtschlüssel (mandatory)
[M/O]	bedingter Pflichtschlüssel
[O]	optionaler Schlüssel (im Kontext der anderen Angaben)

An vielen Stellen dieser Spezifikation werden die Werte bestimmter Schlüssel zur Bildung von Datei– und Verzeichnisnamen benutzt. Um die Nutzung eines Wertes eines Schlüssels zu kennzeichnen, wird dem Schlüssel das Zeichen '$' vorangestellt und die Referenz in runde Klammern eingeschlossen. Das Daten–Stammverzeichnis, welches in Verzeichnisangaben benutzt wird, trägt den symbolischen Namen `<data>`.

2 Verzeichnisstruktur

2.1 Struktur der OFML–Produkt- und Katalogdaten

Die unten stehende Darstellung zeigt eine Übersicht der in pCon–Applikationen verwendeten Verzeichnisstruktur. In den nachfolgenden Kapiteln und Abschnitten wird auf die einzelnen Verzeichnisse und ihren Inhalt näher eingegangen.

```
<data>/
    catalogs/
        images/ ..... [3.1.3]
        profiles/ ..... [3.1.2]
    registry/ ..... [5.2]
    ($manufacturer)/
        ($program)/
            ($version)/ ..... [6.1]
            ($region)/
                ($version)/
                    cat/ ..... [6.3]
                    db/ ..... [6.2]
                    etc/ ..... [6.4.4]
                    html/ ..... [6.4.3]
                        de/
                        en/
                        fr/
                        ...
                    image/ ..... [6.4.1]
                    mat/ ..... [6.4.2]
                       l/
                       m/
                       s/
                    meta/
                    oam/
                    oap/
        priceprofiles/ ..... [6.6]
            ($ppr_version)/
```

Einige Strukturelemente dürfen auch als **ZIP–Archiv** hinterlegt werden:

```
<data>/
    ($manufacturer)/
        ($program)/
            ($region)/
                ($version)/
                    cat/xcf.zip
                    image/image.zip
                    mat/mat.zip
```

Hinweise zur Benutzung von ZIP-Archiven:
Die ZIP–Archive enthalten die Dateien, so wie sie auch im Verzeichnis selbst enthalten wären (d.h keine zusätzlichen Unterverzeichnisse im Archiv). Eine Ausnahme bildet das `mat.zip`, siehe 6.4.2.
Liegen Dateien sowohl im ZIP–Archiv als auch als Datei im Verzeichnis vor, so wird die Datei im Verzeichnis bevorzugt benutzt.

2.2 Beispielstruktur

```
<data>/
    catalogs/
        images/
            man_de-2011.0.jpg
            man_de-2012.0.jpg
            man_fr-2011.0.jpg
            man_fr-2012.1.jpg
            ...
        profiles/
            man.cfg
            man_de-2011.0.cpr
            man_de-2012.0.cpr
            man_fr-2011.0.cpr
            man_fr-2012.1.cpr
            ...
    man/
        series/
            1/odb.ebase
            2/odb.ebase
            3/odb.ebase
            DE/
                1/
                    cat/xcf.zip
                    db/pdata.ebase
                    etc/artsetup.csv
                    image/image.zip
                    mat/mat.zip
                    oam/oam.ebase
                2/
                    cat/xcf.zip
                    db/pdata.ebase
                    etc/artsetup.csv
                    image/image.zip
                    mat/mat.zip
                    oam/oam.ebase
            FR/
                1/
                    cat/xcf.zip
                    db/pdata.ebase
                    etc/artsetup.csv
                    image/image.zip
                    mat/mat.zip
                    oam/oam.ebase
                3/
                    cat/xcf.zip
                    db/pdata.ebase
                    etc/artsetup.csv
                    image/image.zip
                    mat/mat.zip
                    oam/oam.ebase
        ...
    registry/
        man/
            man.jpg
        MAN.cfg
        man_series_DE_1.cfg
        man_series_DE_2.cfg
        man_series_FR_1.cfg
        man_series_FR_3.cfg
        ...
```

3 Daten- und Katalogprofile

Dieses Kapitel beschreibt die Anmeldung von OFML–Daten in einer pCon–Applikation.

3.1 Verzeichnisstruktur der Datenregistrierung

Alle Anwendungssysteme müssen eine feste Verzeichnisstruktur unterstützen, die wie folgt aufgebaut ist:

Verzeichnis (relativ)	Kurzbezeichnung
`/bin`	Ausführbare Programmdateien, System-Bibliotheken
`/lib`	Modulverzeichnis (optional)
`/data`	OFML–Produkt- und Katalogdaten
`/data/catalogs`	Katalogprofile und zugehörige Ressourcen
`/etc/data`	Datenprofilverzeichnis
`/etc/startup`	Konfigurationsdateien

Diese Verzeichnisse befinden sich üblicherweise unterhalb des Programmverzeichnisses.
Einzelne Verzeichnisse können sich aber auch außerhalb des Programmverzeichnisse befinden, um z.B. Server–Lösungen zu realisieren.

OFML–Daten werden über Daten- oder Katalogprofile angemeldet:

- Ein Datenprofil beschreibt einen einzelnen Datenstand eines Herstellers ohne spezifische Kennung für Vertriebsgebiet bzw. Version.
  Datenprofile werden verwendet, wenn im Anwendungssystem nur ein Datenstand eines Herstellers verarbeitet werden soll bzw. kann.
- Ein Katalogprofil beschreibt einen Datenstand (Katalog) eines Herstellers bzw. Lieferanten mit Kennung für Vertriebsgebiet und Version.
  Katalogprofile ermöglichen die gleichzeitige Verwendung von verschiedenen Datenständen eines Herstellers/Lieferanten, die sich in Vertriebsgebiet und Version unterscheiden[1]. Dies wird jedoch (noch) nicht von allen Anwendungssystemen unterstützt. Für die anderen Anwendungssysteme sollte deswegen immer auch noch ein Datenprofil[2] bereitgestellt werden.

Die Angaben der zu verarbeitenden Daten- bzw. Katalogprofile befinden sich in der Startdatei `default.cfg` im Verzeichnis `etc/startup`.

Diese Datei enthält neben einigen systemspezifischen Informationen den Schlüssel *app.gf.data.profile* und/oder den Schlüssel *app.gf.data.catalogs*. Diese Schlüssel listen die Namen der zu ladenden Profile auf.
Alternativ können diese Schlüssel auch aus einer separaten Datei gelesen werden. Diese Datei kann über den Schlüssel *app.gf.data.profile.registration* spezifiziert werden[3].

3.1.1 Registrierung von Datenprofilen via `app.gf.data.profile`

Dieser Schlüssel wird von Anwendungssystemen verwendet, welche die Verarbeitung von Katalogprofilen noch *nicht* unterstützen.

Datenprofile werden als Dateien mit der Endung `.cfg` im Profilverzeichnis bereitgestellt.
Das Profilverzeichnis kann durch den Schlüssel *app.gf.data.profile.path* spezifiziert werden. Dieser Schlüs-

[1] Dieses Konzept ist zuweilen auch unter dem Begriff *Multiple Preislisten* bekannt.
[2] das sogenannte Kompatibilitätsdatenprofil
[3] Der Schlüssel *app.gf.data.profile.registration* hat Vorrang vor den beiden anderen Schlüsseln.

sel ist optional: falls nicht angegeben, wird das lokale Profilverzeichnis (`etc/data`) verwendet. Datenprofile werden bevorzugt aus dem durch *app.gf.data.profile.path* spezifizierten Verzeichnis geladen. Existiert dort keine Profildatei, wird versucht diese aus dem lokalen Verzeichnis zu laden.

Im Schlüssel *app.gf.data.profile* werden die Namen der Profildateien ohne Dateiendung angegeben. Mehrere Profile werden dabei durch ein Semikolon getrennt.

Beispiel 1

```
default.cfg:

app.gf.data.profile = standard_DE_1;he1
```

Beispiel 2

```
default.cfg:

app.gf.data.profile.path = \\SERVER\data\profiles
app.gf.data.profile = standard_DE_1;he1;he2
```

Beispiel 3

```
default.cfg:

app.gf.data.profile.path = \\SERVER\data\profiles
app.gf.data.profile.registration = \\SERVER\data\profiles\app.profiles

\\SERVER\data\profiles\app.profiles:

app.gf.data.profile = standard_DE_1;he1;he2
```

3.1.2 Registrierung von Katalogprofilen via `app.gf.data.catalogs`

Dieser Schlüssel wird von Anwendungssystemen verwendet, welche die Verarbeitung von Katalogprofilen unterstützen.

> Hinweis:
> Sofern der Schlüssel *app.gf.data.catalogs* nicht in der Profilregistrierungsdatei vorhanden ist, liest eine Anwendung die Profile aus dem alten Schlüssel *app.gf.data.profile*. Dies stellt die Abwärtskompatiblität einer Anwendung, die Katalogprofile unterstützt, zu alten Dateninstallationen sicher, die noch nicht migriert wurden. Der alte Schlüssel *app.gf.data.profile* wird jedoch nicht ausgewertet, wenn ein Schlüssel *app.gf.data.catalogs* in der Profilregistrierungsdatei vorhanden ist.

Der Name einer Katalogprofildatei hat die folgende Form:

```
($brand)_($catalog_id).cpr
```

Der Wert des Schlüssels *app.gf.data.catalogs* sind die Dateinamen der zu ladenden Profile *inklusive* Dateiendung aber ohne Pfadangabe. Mehrere Profile sind jeweils mit einem Semikolon separiert.

Neben Katalogprofilen können in dem Schlüssel auch herkömmliche Datenprofile angegeben werden. Damit werden auch Hersteller unterstützt, die noch keine Katalogprofile bereitstellen.

Katalogprofile werden standardmäßig im Profilverzeichnis `catalogs/profiles` des OFML-Datenverzeichnisses gespeichert. Ein alternatives Profilverzeichnis kann über den optionalen Schlüssel `app.gf.data.profile.path` in der Startdatei `etc/startup/default.cfg` eines OFML-Anwendungssystems festgelegt werden. Ist ein Katalogprofil nicht im Profilverzeichnis vorhanden, sucht die Anwendung in ihrem lokalen Pfad `etc/data` danach.

Beispiel

```
app.gf.data.catalogs=man1_de-2012.1.cpr;man2.cfg;man3_any-2012.4.cpr
```

3.1.3 Ablage von Katalogprofil-Ressourcen

Einem Katalogprofil kann ein optionales Logo für die Darstellung in einer Katalogauswahl zugewiesen werden. Dabei können 2 verschiedene Größen angegeben werden:

Katalog–Logo klein:

```
catalogs/images/($brand)_($catalog_id)_.jpg
[catalogs/images/($brand)_($catalog_id)_.png]
```

Bildgröße: [1 - max. 100] x 20 (B x H in Pixel)

Katalog–Logo groß:

```
catalogs/images/($brand)_($catalog_id).jpg
[catalogs/images/($brand)_($catalog_id).png]
```

Bildgröße: [1 - max. 200] x 40 (B x H in Pixel)

Das Grafikdateiformat ist JPEG[4]. Optional kann auch PNG verwendet werden, dessen Verarbeitung ist aber nicht in allen Applikationen garantiert. Liegt ein Logo sowohl im JPEG-Format als auch im PNG-Format vor, hat PNG Vorrang.

Sofern kein Logo für den Katalog hinterlegt ist, verwendet die Anwendung zur Anzeige des Katalogs das Logo der Marke bzw. des Herstellers (s. Abschn. 4.2.3).

3.2 Datenprofile

Datenprofile sind Textdateien, die nach einem festen Schema aufgebaut sind. Die folgende Tabelle gibt einen Überblick über die einzelnen Sektionen:

Sektion	Kurzbezeichnung
`[config]`	Einstellungen und Suchpfade
`[<lang>]`	Sprachspezifische Bezeichnungen
`[lib:`*`$group`*`]`	Pakete einer Gruppe *$group*

Anmerkung:
Der Gruppenbezeichner *$group* sollte das in der Registrierung angegebene Herstellerkürzel verwenden. Es ist zulässig, dieses Kürzel um eine fortlaufende Nummer zu erweitern oder freie Bezeichner zu verwenden, um z.B. herstellerspezifische Pakete in separaten Gruppen zusammenzufassen.

[4] Zu den Bildformat-Konventionen siehe Abschn. 6.5.

3.2.1 Einstellungen und Suchpfade

[1] ***path*** **[M/O]**
– definiert den Paket–Suchpfad

Der Eintrag `path` gibt den Pfad zu den in diesem Profil registrierten Paketen an. Dieser Eintrag muss vorhanden sein, wenn die Pakete nicht im lokalem Datenverzeichnis des Anwendungssystems (`<program>/data`) liegen. Es ist jeweils nur ein Pfad pro Profil erlaubt.
Der Eintrag muss vollständige Pfadangaben der Form `<laufwerk>/<pfad>` oder `//<server>/<pfad>` ohne abschliessenden Slash ('/') enthalten.

[2] ***use_version*** **[O]**
– erlaubt die Verwendung von unversionierten Daten

In der Regel ist der Wert mit `true` vorbelegt. Mit `false` wird die Versionierung deaktiviert. Dies setzt kompatible unversionierte Bibliotheken voraus.

3.2.2 Beschreibungen

Der Name der Sektion `<lang>` muss gemäß ISO–Sprachen–Code (ISO–639-1) angegeben werden.

[3] ***name*** **[O]**
– definiert die Kurzbezeichner des Profiles

[4] ***desc*** **[O]**
– definiert eine ausführliche Beschreibung des Profiles

3.2.3 Pakete einer Gruppe

Über die Einträge der Sektion `[lib:$group]` werden alle vom Anwendungssystem zu benutzenden herstellerspezifischen Pakete angegeben. Jeder Eintrag besteht dabei aus dem Namen der Paket-Registrierungsdatei ohne Datei-Endung `.cfg` (s. Abschn. 5.2) gefolgt vom Gleichheitszeichen und dem Lademodus. Dieser wird entweder mit `true` oder `false` angegeben, wobei `true` das Anwendungssystem anweist, das Paket zu benutzen.

Für eine gegebene OFML-Bibliothek darf nur ein Paket[5] angegeben werden.

[5] für ein definiertes Vertriebsgebiet und eine definierte Version

3.2.4 Beispiel eines Datenprofils: standard_DE_1.cfg

```
[config]
path =

[de]
name = Standard
desc = Profil für EasternGraphics Basisbiblitoheken

[lib:egr]
egr_office_np_DE_1=true
egr_accessories_DE_1=true
```

3.3 Katalogprofile

Ein Katalog ist eine Zusammenstellung von Produkten eines Herstellers bzw. Lieferanten und umfasst aus OFML–Sicht alle OFML–Pakete, die zur Auswahl, grafischen Darstellung, Konfigurierung und Bestellabwicklung der Produkte des Katalogs benötigt werden[6]. Ein Katalog besitzt immer sowohl eine räumliche Dimension (Vertriebsgebiet) als auch eine zeitliche Dimension (Version).

Ein Katalogprofil definiert eine Ausgabe eines Katalogs eines Herstellers/Lieferanten.

3.3.1 Aufbau

Das Katalogprofil wird als eine Konfigurationsdatei gespeichert. Die Einstellungen des Profils sind als Schlüssel-Wert-Paare

<Schlüssel> = <Wert>

in je einer Zeile in verschiedenen Sektionen definiert.

Zeilen, die mit dem Zeichen # beginnen, werden als Kommentar interpretiert.

Im Folgenden werden die Sektionen und Schlüssel einer Katalogprofil-Datei beschrieben.

Sektion	Kurzbezeichnung
[catalog]	Einstellungen und Suchpfade
[*<lang>*]	Sprachspezifische Bezeichnungen
[lib]	vom Katalog benutzte OFML–Pakete

Ein Katalogprofil verwendet UTF-8 als Codepage. Dabei können die ersten 3 Byte das optionale Byte-Order-Mark für UTF-8 enthalten (EF BB BF). Alternativ ist es zulässig, ein Katalogprofil in UTF-16 Little Endian zu speichern. Die Verwendung von UTF-16 ist durch das Byte-Order-Mark FF FE deklariert.

[6] Als Synonym wird zuweilen auch der Begriff *Preisliste* verwendet.

3.3.2 Einstellungen

[5] ***catalog_id*** **[M]**
– gibt den eindeutigen Schlüssel des Katalogs an.

Die Katalog–ID dient der eindeutigen Identifizierung eines Katalogs und der aus ihm in ein OFML–Projekt eingefügten Artikel. Für jedes neue Release eines Katalogs muss daher eine neue Katalog-ID vergeben werden. Es ist nicht zulässig, für zwei verschiedene Versionen eines Katalogs ein und dieselbe Katalog–ID zu verwenden.

Die Katalog-ID hat folgenden Aufbau: `<Identifier>.<Revision>`

Der Identifier identifiziert den Katalog und muss dem regulären Ausdruck `[a-z][a-z0-9_-]*` entsprechen. Es wird die Verwendung eines Identifiers der Form `<Vertriebsgebiet>-<Jahr der Ausgabe>` empfohlen.
Katalog–Revisionen werden verwendet, um Änderungen (Korrekturen) innerhalb eines Katalogs abzubilden. Die Revisionsnummer ist eine einzelne ganze Zahl größer gleich 0. Bei der initialen Ausgabe eines Katalogs ist die Revisionsnummer 0 zu verwenden. Bei jeder ausgelieferten Änderung des Katalogs muss sie entsprechend erhöht werden.

Parallel zu verarbeitende Kataloge müssen unterschiedliche Identifier in ihrer Katalog-ID aufweisen. Ein Revision eines Katalogs ersetzt eine vorherige Revision desselben Katalogs mit niedrigerer Revisionsnummer.

```
Beispiel: catalog_id = de-2011.1
```

[6] ***release_date*** **[M]**
– definiert das Datum an dem der Katalog veröffentlicht wird.

Der Wert dieses Schlüssels muss als ISO-Datum der Form `JJJJ-MM-DD` angegeben werden [iso8601].

```
Beispiel: release_date = 2011-03-17
```

[7] ***brand*** **[M]**
– legt die OFML–Kurzbezeichnung der Marke fest, unter welcher der Katalog geführt wird.

Der Wert des Schlüssels darf nicht frei festgelegt werden. Er muss in der zentralen Registrierungsdatenbank der Hersteller eingetragen sein (s. 4.1).
Wenn der Katalog nicht unter einer speziellen Marke geführt wird, muss hier die Kurzbezeichnung des Herstellers/Lieferanten angegeben werden (Schlüssel *`manufacturer`*, s. 5.3).

[8] ***brand_id*** **[M]**
– legt das eindeutige kaufmännische Kürzel der Marke fest, unter der der Katalog geführt wird.

Der Wert des Schlüssels darf nicht frei festgelegt werdem. Er muss in der zentralen Registrierungsdatenbank der Hersteller eingetragen sein (s. 4.1).
Über diesen Schlüssel werden aus der Herstellerregistrierungsdatei zusätzliche Informationen geladen (z.B. der Anzeigename oder die Herstelleranschrift).
Wenn der Katalog nicht unter einer speziellen Marke geführt wird, muss hier die *`manufacturer_id`* des Herstellers/Lieferanten angegeben werden.

[9] ***distribution_region*** **[M]**
– legt das Basis-Vertriebsgebiet fest, für das der Katalog gültig ist.

Über das Vertriebsgebiet kann eine Anwendung verschiedene Kataloge kategorisieren, um Vorschläge zur Ersetzung anzuzeigen.

[10] ***valid_from*** **[O]**
– definiert das Datum, ab dem der Katalog verwendet werden darf.

Der Wert muss als ISO–Datum `JJJJ-MM-DD` formatiert sein [iso8601].

Ein mit *valid_from* und *valid_to* festgelegter Gültigkeitszeitraum hat lediglich informativen Charakter. Anwendungssysteme schränken die Verwendung des Katalogs nicht auf diesen Zeitraum ein. Diese Information kann jedoch dazu dienen, Warnungen oder Empfehlungen zur Deinstallation bei Überschreitung des Zeitraums einzublenden.

Ist die untere Grenze des Gültigkeitszeitraums nicht festgelegt, gilt das hinreichend weit in der Vergangenheit zurückliegende Datum `1970-01-01`.

[11] ***valid_to*** **[O]**
– legt das Datum fest, bis welches der Katalog verwendet werden kann.

Der Wert muss als ISO–Datum `JJJJ-MM-DD` formatiert sein [iso8601].

Ist die obere Grenze des Gültigkeitszeitraums nicht festgelegt, gilt das hinreichend weit in der Zukunft liegende Datum `9999-12-31`.

[12] ***path*** **[O]**
– definiert den Suchpfad zu den verwendeten OFML-Daten.

Sofern die von einem Katalog verwendeten OFML-Daten nicht im lokalen Datenverzeichnis eines Anwendungssystems liegen, ist der Basispfad zu den Daten über diesen Schlüssel festzulegen.
Die Angabe des Pfades erfolgt in der Form `<laufwerk>/<pfad>` oder `//<server>/<pfad>` ohne abschliessenden Slash ('/').

[13] ***ppr_region_id*** **[O]**
– ordnet den Katalog einer Preisprofilregion zu.

Wenn eine Preisprofilregion aktiviert wird, muss mit dem Katalog auch ein Preisprofilpaket verteilt werden, in dem diese Region definiert ist (s.a. 6.6).

[14] ***ppr_version*** **[M/O]**
– legt die Version des verwendeten Preisprofils fest.

Die Version des verwendeten Preisprofils muss zwingend angeben werden, wenn über den Schlüssel *ppr_region_id* eine Preisprofilregion aktiviert wurde (s.a. 6.6).

```
Beispiel: ppr_version = 1
```

3.3.3 Beschreibungen

Der Name eines Katalogs sowie optionale Beschreibungen werden für eine Standardsprache in der Sektion `[catalog]` angeben.

Darüber hinaus können diese Texte für zusätzliche Sprachen in zusätzlichen optionalen Sektionen `[<lang>]` hinterlegt werden. Der Schlüssel *<lang>* definiert dabei die jeweilige Sprache und muss als zweistelliger ISO-Sprachen-Code (ISO–639-1) angegeben werden [iso639-1].

[15] ***catalog_name*** **[M]**
– legt eine Bezeichnung für den Katalog fest.

Die Bezeichnung soll so gewählt werden, dass sie eine leichte Identifizierung des Katalogs seitens des Anwenders erlaubt z.B. `Katalog <Vertriebsgebiet> <Jahr>/<Ausgabe>`.

Anmerkung:
Kataloge werden in der Regel hierarchisch unterhalb des Herstellers (Lieferanten) aufgelistet. Anwendungen, die eine herstellerübergreifende Liste verfügbarer Kataloge anbieten, können den Herstellernamen mit den Katalognamen verknüpfen. Der Name des Herstellers muss daher nicht zwingend zusätzlicher Bestandteil des Katalognamens sein.

[16] ***description*** **[O]**
– legt eine ausführliche Beschreibung des Profiles fest

Feste Zeilenumbrüche können in der Beschreibung durch `\n` definiert werden.

3.3.4 Datenpakete

Die von einem Katalog verwendeten Pakete werden in der Sektion `[lib]` angegeben. Für jedes Paket ist ein Schlüssel der Form

<Paketschlüssel> = [true|false]

anzugeben. Der *<Paketschlüssel>* entspricht dabei dem Namen der Paket-Registrierungsdatei ohne Dateiendung (s. Abschn. 5.2). Der Wert `true` gibt an, dass das Paket benutzt wird.

Für eine gegebene OFML-Bibliothek darf nur ein Paket[7] angegeben werden.

Kann eine Paket-Registrierungsdatei nicht geladen werden, kann auch der gesamte Katalog nicht geladen/verwendet werden.

Ein Katalogprofil kann Pakete mehrerer Hersteller umfassen, die Pakete mit Katalogdaten müssen jedoch von ein und demselben Hersteller bzw. von Herstellern eines Konzerns (*brand*) oder von den Herstellern `::egr::` und `::ofml::` stammen.

[7] für ein definiertes Vertriebsgebiet und eine definierte Version

4 Registrierung von Herstellern und Konzernen

4.1 Zentrale Registrierungsdatenbank

Hersteller[8] und Konzerne werden bei EasternGraphics zentral registriert.
(Bitte kontaktieren Sie cto@EasternGraphics.com.)
Dies vermeidet Konflikte bei der Vergabe der eindeutigen Schlüssel. Diese Registrierungsinformationen werden in einer globalen Hersteller-Datenbank verwaltet (`Manufacturers.ebase`).

Diese Datenbank beinhaltet die Zuordnung der Kurzbezeichnungen (IDs) von Herstellern und Konzernen sowie deren Namen.

Eine Liste der vergebenen Kurzbezeichnungen kann jederzeit bei EasternGraphics angefordert werden.

Die im Folgenden beschriebenen Schlüssel werden teilweise nicht ausgewertet, wenn eine Anwendung die Informationen direkt aus der globalen Hersteller-Datenbank verwendet.

4.2 Hersteller-Registrierung

Für jeden Hersteller muss eine eigene Registrierungsdatei angelegt werden. In dieser können herstellerspezifische, paketübergreifende Inhalte abgelegt werden.

Der Dateiname wird aus der kaufmännischen Kurzbezeichnung (ID) des Herstellers (laut globaler Hersteller-Datenbank, s.o.) gebildet. Die Dateiendung ist `.cfg`.

Die Hersteller–Registrierungsdateien befinden sich im Verzeichnis `<data>/registry`.

Beispiel:

```
<data>/registry/MAN.cfg
```

4.2.1 Format der Hersteller-Registrierung

Das Format der Hersteller–Registrierungsdateien gleicht dem der Paket-Registrierungsdateien (s. 5.1). Die Datei ist in zwei Bereiche gegliedert, eine sprachunabhängige `[general]` Sektion und in mehrere sprach–spezifische `[<ISO-Kürzel>]` Sektionen. Jeder Schlüssel kann prinzipiell sowohl in der Sektion `[general]` als auch in sprach-spezifischen Sektionen verwendet werden. Das Anwendungssystem sucht einen Schlüssel zuerst immer in der relevanten sprach-spezifischen Sektion.

Jeder Schlüssel aus der Paket–Registrierung kann prinzipiell auch in der Hersteller–Registrierung spezifiziert werden[9]. Dieser wird immer dann benutzt, wenn dieser Schlüssel in der Paket–Registrierung nicht hinterlegt ist.

[8] Als *Hersteller* wird hier im Allgemeinen jede Instanz bezeichnet, die OFML–Daten bereitstellt und verteilt. Neben eigentlichen Herstellern von Produkten kann dies z.B. also auch ein Verband o.ä. sein.

[9] auch wenn das bei bestimmten Schlüsseln praktisch nicht sinnvoll ist

Beispiel:

```
[general]
address.name = EasternGraphics GmbH
address.street = Einsteinstrasse 1
address.city = Ilmenau
address.zip = 98693
address.country = Deutschland
address.tel = 03677 6782-0
address.fax = 03677 678250
address.email = info@EasternGraphics.com
address.www = www.EasternGraphics.com
ppr_region_id = GER

[de]
manufacturer_name = Standard

[en]
manufacturer_name = Default
address.country = Germany
```

4.2.2 Schlüssel der Hersteller-Registrierung

[1] ***concern_id*** **[O]**
– definiert die eindeutige Kurzbezeichnung des Konzerns, dem der Hersteller angehört

Der Bezeichner darf alphanumerische Zeichen inklusive dem Unterstrich enthalten, wobei das erste Zeichen ein Buchstabe sein muss:

`[a-zA-Z][a-zA-Z0-9_]*`

Achtung: Dieser Schlüssel muss registriert werden (s. 4.1).

Hinweis:
Konzernbezeichner dürfen sowohl Groß- als auch Kleinbuchstaben enthalten, die korrekte gleiche Schreibweise muss jedoch bei jeder Verwendung des Bezeichners eingehalten werden. Es dürfen keine zwei Konzernbezeichner existieren, die sich nur in der Schreibweise der Buchstaben unterscheiden.

[2] ***manufacturer_name*** **[M]**
– definiert die (sprachabhängige) Bezeichnung des Herstellers

[3] ***ppr_region_id*** **[O]**
– bestimmt den im Preisprofil des Herstellers zu verwendenden Gebietsschlüssel

Wird im Preisprofil in der Spalte $country$ der Tabelle $profile$ referenziert. (s. [ppr]). (Siehe auch 6.6)

`Bsp.: ppr_region_id = BENELUX`

[4] ***address.name*** **[O]**
[5] ***address.street*** **[O]**
[6] ***address.postbox*** **[O]**
[7] ***address.city*** **[O]**
[8] ***address.zip*** **[O]**
[9] ***address.state*** **[O]**
[10] ***address.country*** **[O]**
[11] ***address.tel*** **[O]**
[12] ***address.fax*** **[O]**
[13] ***address.email*** **[O]**
[14] ***address.www*** **[O]**

Auf eine nähere Beschreibung der Adress–Schlüssel wird an dieser Stelle verzichtet, da deren Bedeutung aus den Bezeichnungen im oberen Beispiel hervorgeht.

Eine Besonderheit bei diesen Schlüsseln besteht darin, dass die Werte auch mehrzeilig hinterlegt werden können. Dazu muss dem Schlüssel ein Index als Postfix angehängt werden. Dabei bestimmt die Folge der Indizes und nicht die Reihenfolge der Zeilen die Zusammensetzung des Textes.

[15] ***sort_names*** **[O]**
– erzwingt die alphanummerische Sortierung der Serien nach den lokalisierten Seriennamen in der Darstellung innerhalb des Anwendungssystems

Bei Belegung mit *false* (Standard) findet keine Sortierung statt, die Serien werden in ihrer im Datenprofil (s. 3.2 auf Seite 8) angegebene Reihenfolge angezeigt.

```
Bsp.: sort_names = true
```

[16] ***release_note*** **[O]**
– definiert einen beliebigen Text zum Hersteller

Dieser Text kann paketübergreifende Informationen enthalten. Ein Anwendungssystem kann diesen Text z.B. in der Katalogauswahl anzeigen.

```
Bsp.: release_note = Version 2.3 mit Preisanpassung vom 01.04.
```

[17] ***gln_id*** **[O]**
– gibt die Global Location Number (GLN) des Herstellers / Konzerns an (s. [gln])

[18] ***series_name.<program_id>*** **[O]**
– Angabe des Namens der kfm. Serie *program_id*

Der Name kann z.B. vom Anwendungssystem in Auswahldialogen verwendet werden.

Ist dieser Schlüssel für eine gegebene kfm. Serie *program_id* nicht hinterlegt, so muss das Anwendungssystem versuchen, anhand von kfm. Hersteller- und Serien–ID die passende OFML–Bibliothek zu ermitteln, um den Namen aus deren Registrierungsdatei zu verwenden (Schlüssel *program_name*). Dieses Verfahren ist aber nicht zuverlässig, da nicht immer eine eindeutige Beziehung zwischen kfm. Serie und OFML–Bibliothek besteht.

```
Bsp.: series_name.EX = Example
```

[19] ***external_catalog.url* [O]**
[20] ***external_catalog.name* [O]**

Der Schlüssel *external_catalog.url* darf nur einmal in der Sektion [general] vorkommen. Wenn der Wert des Schlüssels eine gültige HTTPS-URL ist, werden in den OFML-Paketen des Herstellers hinterlegte Katalogdaten ignoriert und die Applikation bietet stattdessen eine Verbindung zu dem durch die URL definierten **externen Katalog** an.

Der Schlüssel *external_catalog.name* kann verwendet werden, um sprach-spezifische Bezeichnungen für den Katalog zu hinterlegen. Existiert für die in der Applikation eingestellte Katalog-Sprache kein Schlüssel in der entsprechenden sprach-spezifischen Sektion, wird der Schlüssel aus der Sektion [general] verwendet. Existiert der Schlüssel auch dort nicht, wird der Name des Herstellers verwendet (siehe Schlüssel *manufacturer_name* oben).

Externe Kataloge werden nicht von allen pCon–Applikationen unterstützt. Damit Artikel aus dem externen Katalog in ein Projekt eingefügt werden können, müssen die externen Kataloge ein spezielles, applikationsspezifisches API bedienen[10].

Beispiel:

```
[general]
external_catalog.url=https://www.example.com
external_catalog.name=Example Catalog

[de]
external_catalog.name=Beispielkatalog
```

4.2.3 Ablage von Hersteller-Ressourcen

Herstellerspezifische Ressourcen werden im Verzeichnis <data>/registry/($manufacturer) abgelegt.

Folgende Ressourcen sind definiert:

Herstellerlogo:

```
registry/($manufacturer)/($manufacturer).jpg
[registry/($manufacturer)/($manufacturer).png]
```

- Bildgröße: [1 - max. 200] x 40 (Breite x Höhe in Pixel)

großes Herstellerlogo (optional):

```
registry/($manufacturer)/($manufacturer)_l.jpg
[registry/($manufacturer)/($manufacturer)_l.png]
```

- Das Bild darf eine maximale Kantenlänge von 2000 Pixeln haben.
- Die längste Kante sollte als Empfehlung mindestens 1000 Pixel lang sein.
- Es gibt keine Vorgabe zum Seitenverhältnis.
- Die Anwendungen sind dafür zuständig, Bilder mit unterschiedlichen Größen korrekt zu verarbeiten.

Die unterstützten Grafikdateiformate sind JPEG oder PNG[11]. Liegt ein Logo sowohl im JPEG-Format als auch im PNG-Format vor, hat PNG Vorrang.

[10] Details hierzu können bei den Produktverantwortlichen oder dem Support erfragt werden.
[11] Zu den allgemeinen Bildformat-Konventionen siehe Abschn. 6.5.

4.3 Konzern-Registrierung

Die Konzern–Registrierung funktioniert analog zur Hersteller–Registrierung. Sie ist der Hersteller–Registrierung vorgeschaltet, wie die Hersteller–Registrierung der Paket–Registrierung.

Für jeden Konzern kann eine eigene Registrierungsdatei angelegt werden. Darin können konzernspezifische Inhalte abgelegt werden.

Der Dateiname wird aus der Kurzbezeichnung (ID) des Konzerns (laut globaler Hersteller-Datenbank, s. 4.1) gebildet. Die Dateiendung ist `.cfg`.

Die Konzern–Registrierungsdateien befinden sich im Verzeichnis `<data>/registry`.

4.3.1 Format der Konzern-Registrierung

Das Format der Konzern–Registrierungsdateien ist weitgehend identisch zur Hersteller–Registrierung. Jeder Schlüssel aus der Hersteller– bzw. Paket–Registrierung kann auch in der Konzern–Registrierung spezifiziert werden. Diese werden immer dann benutzt, wenn ein Schlüssel in der Hersteller– bzw. Paket–Registrierung nicht hinterlegt ist.

Beispiel:

```
[general]
concern_id=egr

[de]
concern_name = EasternGraphics GmbH
```

4.3.2 Schlüssel der Konzern-Registrierung

Ergänzend zu den Schlüssel für die Hersteller–Registrierung sind folgende Schlüssel explizit für die Konzern–Registrierung definiert:

[1] ***concern_name*** **[M]**
– definiert die (sprachabhängige) Bezeichnung des Konzerns

4.3.3 Ablage von Konzern-Ressourcen

Konzernspezifische Ressourcen werden im Verzeichnis `<data>/registry/($concern)` abgelegt.

Folgende Ressourcen sind definiert:

Konzernlogo:

```
registry/($concern)/concern.jpg
[registry/($concern)/concern.png]
```

- Bildgröße: [1 - max. 200] x 40 (Breite x Höhe in Pixel)

großes Konzernlogo (optional):

```
registry/($concern.)/($concern.)_l.jpg
[registry/($concern.)/($concern.)_l.png]
```

- Das Bild darf eine maximale Kantenlänge von 2000 Pixeln haben.
- Die längste Kante sollte als Empfehlung mindestens 1000 Pixel lang sein.
- Es gibt keine Vorgabe zum Seitenverhältnis.
- Die Anwendungen sind dafür zuständig, Bilder mit unterschiedlichen Größen korrekt zu verarbeiten.

Die unterstützten Grafikdateiformate sind JPEG oder PNG[12]. Liegt ein Logo sowohl im JPEG-Format als auch im PNG-Format vor, hat PNG Vorrang.

[12] Zu den allgemeinen Bildformat-Konventionen siehe Abschn. 6.5.

5 Registrierung der OFML–Paketen

Für jedes Paket, das in einer pCon–Applikation verwendet werden soll, muss eine Registrierungsdatei angelegt werden. Über diesen Weg werden die Inhalte, die Struktur und die Randbedingungen des Pakets festgelegt.

Zur Ablage der Dateien siehe Abschn. 2.1 und zur Einbindung in Daten- bzw. Katalogprofile siehe Abschnitte 3.2 bzw. 3.3.

5.1 Format der Paket–Registrierung

Das Format der Registrierungsdatei besteht aus (sprach-spezifischen) Sektionen, Schlüsseln und den dazugehörigen Werten. Leerzeilen und Kommentare (eingeleitet mit einem Doppelkreuz '#') sind zulässig. Sprachabhängige Werte sind in eigenen Sektionen abgelegt. Der Sektionsname ergibt sich aus dem zweistelligen ISO–Sprachen–Code (ISO–639-1) (s. 5.4). Die Bedeutung der Schlüssel und Werte wird im nächsten Abschnitt erläutert.

5.2 Registrierung

Im Daten–Stammverzeichnis befindet sich das Verzeichnis `<data>/registry`. In diesem Verzeichnis werden die Registrierungsdateien abgelegt. Der Name einer Paket–Registrierungsdatei wird aus den Schlüsseln gebildet, wie sie in der Datei eingetragen sind. Für den Dateinamen gilt folgende Bildungsvorschrift[13]:

```
($manufacturer)_($program)_($region)_($version).cfg
```

Die Paket–Registrierungsdateien werden über ihren Dateinamen in den Datenprofilen (s. Abschn. 3.2) bzw. Katalogprofilen (s. Abschn. 3.3) referenziert.

5.3 Schlüssel der Paket–Registrierung

[1] ***encoding*** **[O]**
– gibt die Kodierung der Registrierungsdatei und des Pakets selbst an

Dieser Schlüssel sollte nur spezifiziert werden, wenn alle Daten des Pakets die angegebene Zeichenkodierung aufweisen. Die Applikationen versuchen dann, die Daten entsprechend zu dekodieren, unabhängig von der Kodierung, die durch die System-Locale festgelegt ist.

Mögliche Werte sind:

Kodierung	Hinweis
ISO-8859-1	ISO–Latin-1 (Standardwert)
UTF-8	nach Standard ISO/IEC 10646-1 (UCS Transformation Format 8 Bit)

```
Bsp.: encoding = ISO-8859-1
```

[13] Umgekehrt können die Werte für `program` und `region` nicht eindeutig durch Zeichenkettenzerlegung aus dem Namen einer Paket–Registrierungsdatei bestimmt werden (da OFML-Programmkürzel selber einen Unterstrich enthalten können).

[2] ***manufacturer*** **[M]**
– definiert die eindeutige OFML-Kurzbezeichnung des Herstellers

Der Bezeichner darf alphanumerische Zeichen enthalten, wobei Buchstaben klein geschrieben sein müssen und das erste Zeichen ein Buchstabe sein muss:

```
[a-z][a-z0-9]*
```

Achtung: Dieser Schlüssel muss registriert werden (s. 4.1).

Die Kurzbezeichnung des Herstellers bildet den Verzeichnisnamen der ersten Ebene der in Abschnitt 2.1 beschriebenen Verzeichnisstruktur.

```
Bsp.: manufacturer = ofml
```

[3] ***program*** **[M]**
– definiert die eindeutige OFML-Kurzbezeichnung der OFML–Bibliothek (Serie) innerhalb des Herstellers

Der Bezeichner darf alphanumerische Zeichen inklusive dem Unterstrich enthalten, wobei das erste Zeichen ein Buchstabe sein muss:

```
[a-zA-Z][a-zA-Z0-9_]*
```

Die Kurzbezeichnung der Serie bildet den Verzeichnisnamen der Ebene unterhalb des Herstellers. (Verzeichnisstruktur siehe Abschnitt 2.1)

```
Bsp.: program = goiex
```

[4] ***manufacturer_id*** **[M/O]**
– definiert die eindeutige kfm. Kurzbezeichnung des Herstellers

Der Bezeichner darf alphanumerische Zeichen enthalten, wobei Buchstaben groß geschrieben sein müssen:

```
[A-Z0-9]*
```

Achtung: Dieser Schlüssel muss registriert werden (s. 4.1).

Pflichtschlüssel wenn eine Hersteller–Registrierungdatei existiert und für Pakete des Typs *product*. Jedes Kürzel muss einheitlich in allen Paketen eines Herstellers verwendet werden. Eine eineindeutige Zuordnung *manufacturer_id* $\leftrightarrow$ *manufacturer* muss gegeben sein.
Auch in referenzierenden Paketen darf nur ein Kürzel verwendet werden. Das gleichzeitige Referenzieren mehrerer Hersteller ist nicht möglich.

```
Bsp.: manufacturer_id = EG
```

[5] ***program_id*** **[M/O]**
– definiert die (innerhalb des Herstellers) eindeutigen Kurzbezeichnungen der mit diesem Paket abgebildeten kfm. Serien[14]

Pflichtschlüssel für Pakete des Typs *product*. Für Pakete des Typs *catalog* müssen hier alle kfm. Serienkürzel der referenzierten Pakete angegeben werden, sofern sie nicht schon in der Registrierungsdatei dieser Pakete angegeben sind.

[14] Definition s. [glos]

Serienkürzel dürfen nicht mehr als 16 Zeichen enthalten. Idealerweise sollten nur die Großbuchstaben und Ziffern aus dem ASCII–Zeichensatz sowie der Unterstrich '_' verwendet werden.
Die Verwendung folgender Zeichen wird *nicht* empfohlen: `\/?:*"><|,;=` sowie das Leerzeichen[15].

Jedes Kürzel sollte in allen Serien nur genau einmal auftreten, um ein eindeutiges Rückmapping *program_id* → *program* zu ermöglichen. Mehrere Kürzel sind durch Semikolon zu trennen. Die Kürzel müssen exakt so angegeben werden, wie in den kaufmännischen Daten hinterlegt.

Werden kfm. Serien über mehrere OFML–Pakete verteilt, müssen folgende Voraussetzungen erfüllt sein:

a) die Artikelnummern sind herstellerweit eindeutig (kein wiederholtes Vorkommen in einer anderen Serie)

   UND

b) die Artikelnummern müssen im Katalog der jeweiligen OFML–Bibliothek enthalten sein

```
Bsp.: program_id = GX
```

[6] ***distribution_region*** **[M/O]**
– definiert die Kurzbezeichnung des Vertriebsgebietes für das zu registrierende Paket

Vertriebsgebiete stellen eine logische Separierung für die Ablage der Kataloge und kfm. Daten dar. Mögliche Bezeichner für Vertriebsgebiete sind der Landes–Code gemäß ISO–3166-1 Alpha-2 bzw. Alpha-3 (s. [iso3166]) oder ein freier Bezeichner, der nur die Zeichen [a-z, A-Z, 0-9] beinhalten darf und mit einem der erlaubten Buchstaben beginnt. Die Kurzbezeichnungen der Vertriebsgebiete bilden die Verzeichnisnamen der Ebene unterhalb der Serie (s. 2.1). Pflichtschlüssel für Pakete des Typs *product* und *catalog*.

```
Bsp.: distribution_region = DE
```

[7] ***release_version*** **[M]**
– definiert die Versionsnummer des registrierten Pakets

Für den Aufbau der Versionsnummer gibt es unterschiedliche Vorgaben abhängig davon, ob das Paket im Datenprofil oder in Katalogprofilen des Herstellers referenziert wird (s. Abschnitte 3.2 bzw. 3.3):

- Wird das Paket in einem Datenprofil referenziert, so ist die Versionsnummer in der Form `Major.Minor.Build` anzugeben.
  Die Teile sind ganze Zahlen. Major–Nummern beginnen bei 1, die anderen bei 0.
  Für die Inkrementierung der Versionsnummer bei Änderungen in den Produktdaten gelten folgende Empfehlungen:

  Build — Die Inkrementierung der Build–Nummer bedeutet eine *minimale* Änderung im Sinne von Fehlerbeseitigungen (Grafik, Preis, etc.).

  Minor — Die Inkrementierung der Minor–Nummer bedeutet darüber hinaus gehende Änderungen, z.B. der Wegfall oder das Hinzufügen von Artikeln, Merkmalen, Merkmalswerten, etc. Auch ein reguläres Preislisten–Update erfordert ein Minor–Inkrement.

  Major — Die Inkrementierung der Major–Nummer bedeutet gravierende Änderungen, wie z.B. inkompatible Änderungen in den grafischen Daten.

  Wird das Paket auch in Katalogprofilen referenziert[16], so ist nur die Major–Nummer relevant, d.h., die Minor- und Build–Nummern sind immer 0.

- Wird das Paket ausschließlich in Katalogprofilen referenziert, so ist die Versionsnummmer eine einzelne ganze Zahl größer 0 (d.h. besteht nur aus einer Major–Nummer).

[15] Werden diese Zeichen verwendet, ist die korrekte Verarbeitung der Daten auf allen Plattformen und in allen Applikationen der Eastern-Graphics GmbH *nicht* gewährleistet.
[16] Bei dem Datenprofil handelt es sich in dem Fall um das sogenannte Kompatibilitätsdatenprofil.

In beiden Fällen muss die Versionsnummer vor einer Freigabe eines geänderten OFML–Paketes inkrementiert werden (d.h., die Versionsnummer ist streng monoton wachsend).
Eine Installationsroutine für OFML–Pakete kann dann mit dieser Versionsnummer entscheiden, ob eine Bibliothek aktualisiert werden muss.

Die Versionsnummer wird unabhängig vom Vertriebsgebiet vergeben. Somit kann es innerhalb eines Herstellers mehrere Pakete mit identischer Versionsnummer geben, welche sich dann aber im Vertriebsgebiet unterscheiden müssen.

```
Bsp.: release_version = 1.3.1
```

[8] ***release_date*** **[M]**
– definiert das Datum der Veröffentlichung des Pakets

Das Datum ist gemäß ISO–Datumsformat (s. [iso8601]) anzugeben: `YYYY-MM-DD`

```
Bsp.: release_date = 2004-01-17
```

[9] ***release_timestamp*** **[O]**
– gibt den finalen Erstellungszeitpunkt eines Pakets an.

Die Zeit wird dabei auf UTC (s. [utc]). normiert und im folgenden Format angegeben: `yyyyMMddhhmmss`

```
Bsp.: release_timestamp = 20100204145736
```

Es wird empfohlen, den Schlüssel *`release_timestamp`* anzugeben, da es den Applikationen damit – zusammen mit den Schlüsseln *`release_version`* und *`release_date`* – ermöglicht wird, sicher zu erkennen, ob eine neue Version des Pakets installiert ist[17].

[10] ***release_state*** **[O]**
– gibt den (Release–) Status des Pakets an

Mögliche Werte sind: *`alpha, beta, rc, test, final`*. Für alle nicht leeren Werte verschieden von *`final`* und *leer* gibt das Anwendungssystem eine Hinweismeldung (ggf. *`release_text`*).

```
Bsp.: release_state = final
```

[11] ***release_text*** **[O]**
– definiert einen ggf. vom Anwendungssystem auszugebenden Status-Text

(siehe *`release_state`*).

```
Bsp.: release_text = Non official test version!
```

[12] ***version*** **[M]**
– definiert die Hauptversionsnummer des registrierten Pakets

Diese Version muss mit der Major–Nummer des Schlüssels *`release_version`* identisch sein.
Die Versionsnummer bildet eine Namensraum–Ebene innerhalb des OFML–Archivs und ein Verzeichnis unterhalb des Vertriebsgebiets. Beide müssen synchron innerhalb einer Bibliothek bezeichnet werden. (s. 2.1)

[17] im Vergleich zum Zeitpunkt der letzten Speicherung bzw. Aktualisierung eines Projektes

```
Bsp.: version = 1
```

[13] ***languages*** **[M]**
– definiert die von dem Paket unterstützten Sprachen

Alle Sprachen werden gemäß ISO–Sprachen–Code (ISO–639-1) zweistellig in Kleinbuchstaben angegeben. Mehrere Sprachen sind durch Semikolon zu trennen.
Wird vom Anwendungssystem eine nicht im Paket unterstützte Sprache angefordert, wird auf die Standardsprache zurückgegriffen. Standardsprache ist die zuerst in der Liste aufgeführte Sprache.

```
Bsp.: languages = de;en;fr;nl;es
```

[14] ***type*** **[M]**
– legt den Typ des Pakets fest

Folgende Typen können verwendet werden:

Typ	Beschreibung
found	OFML–Basisbibliothek ohne Produkt– und Katalogdaten
product	OFML–Produktbibliothek mit Produktdaten und optional Katalogdaten
catalog	Katalogdaten ohne eigene Produktdaten
extension	nachladbare Erweiterungsbibliothek

```
Bsp.: type = found
```

[15] ***productdb*** **[O]**
– enthält den Klassennamen der Produktdatenbank

Es ist der voll qualifizierte OFML–Klassenname (s. [ofml]) anzugeben.

```
Bsp.: productdb = ::ofml::xoi::xOiOCDProductDB21
```

[16] ***productdb_path*** **[O]**
– definiert den Pfad zu den Dateien der Produktdatenbank

Der Pfad wird als Verzeichnisname relativ zum Daten–Stammverzeichnis angegeben. Als Trennzeichen für die Verzeichnisse im Verzeichnisnamen ist immer das Slash–Zeichen ('/') zu verwenden.

```
Bsp.: productdb_path = ofml/goiex/DE/1/db
```

[17] ***oam_path*** **[O]**
– definiert den Pfad zur OAM–Datenbank

Dieser Schlüssel wird analog zu *productdb_path* angegeben.
Standardmäßig: `<productdb_path>/../oam`

```
Bsp.: oam_path = ofml/goiex/DE/1/oam
```

[18] ***mddb_path*** **[O]**
– definiert den Pfad zur MetaDialog–Datenbank

Diese befindet sich standardmäßig unter:

```
<data>/($manufacturer)/($program)/($version)
```

Dieser Schlüssel wird analog zu *productdb_path* angegeben.

```
Bsp.: mddb_path = ofml/goiex/1/md
```

[19] ***proginfo*** **[O]**
– definiert den Namen der Programm–Info–Klasse

Es ist der voll qualifizierte OFML–Klassenname (s. [ofml]) anzugeben.

```
Bsp.: ::ofml::goiex::gOiExProgInfo
```

[20] ***proginfodb_path*** **[O]**
– definiert den Pfad zur Programm–Info–Datenbank

Diese befindet sich standardmäßig unter:

```
<data>/($manufacturer)/($program)/($version)
```

Dieser Schlüssel wird analog zu *productdb_path* angegeben. Er ist nur wirksam, wenn sich im Standard-Pfad keine Info-DB befindet (d.h. eine Info-DB im Standard-Pfad der Serie hat Vorrang vor einer evtl. vorhandenen DB im proginfodb_path).

```
Bsp.: proginfodb_path = ofml/oi/1
```

[21] ***category*** **[M/O]**
– legt eine Kategorie für die Bibliothek fest

Die Kategorie legt fest, in welche Planungshierarchie die Objekte der Bibliothek geplant werden. Pflichtschlüssel für Pakete des Typs *product*.

Kategorie	Beschreibung
furniture	Möbel und Gegenstände, welche mit Möbeln geplant werden
building	Raum– und Gebäudeelemente
undefined	Basis- oder nicht zuordenbare Bibliothek

```
Bsp.: category = furniture
```

[22] ***depend*** **[M/O]**
– definiert die Pakete, die zum Laden des vorliegenden Pakets Voraussetzung sind (Abhängigkeiten)

Mehrere Pakete werden durch ein Semikolon getrennt.
Die Liste sollte möglichst minimal sein, d.h. nur Pakete enthalten, die von dem vorliegenden Paket direkt benötigt werden. Es ist darauf zu achten, dass die angegebenen Pakets ebenfalls Abhängigkeiten besitzen können (Gefahr von zirkulären Abhängigkeiten).

Schlüssel-Syntax: `::($manufacturer)::($program)::($release_version)/($distribution_region);...`

Die Angabe der Minor- und Build–Nummer in der Release-Version ist optional.

Die Release-Version wird durch die Applikation geprüft, wenn das vorliegende Paket aus einem Datenprofil geladen wurde, oder wenn das vorausgesetzte Paket nicht in dem Katalogprofil aufgelistet ist, aus dem das vorliegende Paket geladen wurde[18].

Bei Katalogprofilen wird die Einhaltung der Abhängigkeiten zwischen den darin aufgelisteten Paketen vorausgesetzt. Den Applikationen, die Katalogprofile verarbeiten, ist freigestellt, ob sie dennoch eine Prüfung der Abhängigkeiten vornehmen. Alternativ kann auch eine Prüfung durch die Installationsroutine stattfinden.

Bei Prüfung der Abhängigkeiten durch die Applikation müssen folgende Bedingungen erfüllt sein, damit das vorausgesetzte Paket in der aktuell installierten Version verwendet werden kann:

1. Ist das vorausgesetzte Paket nicht in dem Daten- bzw. Katalogprofil aufgelistet, muss die Major-Nummer der installierten Version exakt mit der in diesem Schlüssel genannten Major-Nummer übereinstimmen, ansonsten muss die Major-Nummer der installierten Version größer gleich der in diesem Schlüssel genannten Major-Nummer sein.
   Sind beide Major-Nummern identisch, wird, falls in diesem Schlüssel angegeben, noch die Minor-Nummer geprüft:
2. Die Minor-Nummer der installierten Version muss größer gleich der in diesem Schlüssel genannten Minor-Nummer sein.
   Sind beide Minor-Nummern identisch, wird, falls in diesem Schlüssel angegeben, noch die Build-Nummer geprüft:
3. Die Build-Nummer der installierten Version muss größer gleich der in diesem Schlüssel genannten Build-Nummer sein.

```
Bsp.: depend = ::ofml::goi::1.2.0/ANY;::ofml::xoi::1.11.10/ANY
```

[23] ***catalogs*** **[M/O]**
– gibt die Bibliotheken an, die über ihren Katalog Artikel der vorliegenden Bibliothek referenzieren

Wenn Artikel des vorliegenden Pakets in den Katalogdaten anderer Bibliotheken referenziert werden, so *müssen* diese Bibliotheken in diesem Schlüssel angegeben werden, anderenfalls kann in bestimmten Anwendungsszenarien der für die Einfügung eines Artikel erforderliche Katalogeintrag nicht ermittelt werden oder die Suche im Katalog kann fehlschlagen.

Mehrere Bibliotheken werden durch ein Semikolon getrennt.

Schlüssel-Syntax: `::($manufacturer)::($program)::;...`

Von der Applikation werden die Pakete der angegebenen OFML-Bibliotheken verwendet, welche in demselben Daten- bzw. Katalogprofil gelistet sind, aus dem auch das vorliegende Paket geladen wurde.

```
Bsp.: catalogs = ::egr::office::;::egr::office_np::
```

[24] ***oap_program*** **[M/O]**
– gibt die Bibliothek an, die die OAP-Daten für die Artikel der vorliegenden Bibliothek enthält

Fehlt der Schlüssel, werden die OAP-Daten [oap] in der vorliegenden Bibliothek selber erwartet.

Schlüssel-Syntax: `::($manufacturer)::($program)::`

[18] Letzteres trifft normalerweise nur für OFML–Basisbibliotheken zu, da diese nicht von dem Hersteller selber bereitgestellt werden.

Von der Applikation wird das Paket der angegebenen OFML-Bibliothek verwendet, das in demselben Daten- bzw. Katalogprofil gelistet ist, aus dem auch das vorliegende Paket geladen wurde.

[25] ***pd_format*** **[M/O]**
– gibt das verwendete Produktdatenformat an

Die Angabe dieses Schlüssels ist Pflicht für Pakete, die kfm. Daten beinhalten. Alle anderen Pakete sollten explizit `NULL` angeben.

Folgende Formate sind definiert:

- *OCD_2.1*, *OCD_2.2*, *OCD_3.0*, *OCD_4.0*, *OCD_4.1*, *OCD_4.2*, *OCD_4.3*
- *NULL* (Keine kfm. Produktendaten)

```
Bsp.: pd_format = OCD_4.3
```

[26] ***meta_type*** **[O]**
– gibt an, ob eine Instanziierung im Rahmen eines Meta-Typs möglich ist

Ist dieser Schlüssel angegeben und nicht leer und ist eine entsprechende DLM–Lizenz für den Hersteller der Serie vorhanden, werden Artikel durch einen Meta–Typ gekapselt erzeugt.

Es wird die im zweiten Teil angegebene Funktion gerufen, um den zugehörigen Meta-Typ zu ermitteln. Die Angabe des Typs vor der Angabe der Funktion ist notwendig, um durch Evaluierung das Laden des OFML-Klassen Dateien sicherzustellen. Die angegebene Funktion muss also auch in der Klassendatei des angegebenen Typs implementiert sein, oder in Klassen die implizit durch diese Klasse geladen werden.

```
Bsp.: meta_type = ::ofml::go::GoMetaType;::ofml::go::goGetMetaType()
```

[27] ***features*** **[O]**
– definiert spezielle für dieses Paket freigeschaltete Funktionen

Mehrere Funktionen können durch Semikolons getrennt angegeben werden.

Folgende Funktionsschlüssel sind verfügbar:

Funktion	Beschreibung
showInitProgressDlg	Anzeige eines Forstschrittdialoges beim Initialisieren der Produktdatenbank des Pakets.
denyFreePos	Freies Positionieren der Elemente für dieses Paket unterbinden.
deny3DFdbMove	Freies Verschieben der Elemente im 3D für dieses Paket unterbinden.
editableCatalog	Erlaubt die Bearbeitung des Kataloges durch den Anwender zur Laufzeit.
programProperties	Gibt an, dass das Paket statische Merkmale am ProgInfo-Objekt hinterlegt.
useMetaDialogs	Gibt an, dass für das Paket spezielle Konfigurationsdialoge verfügbar sind.
maySetFinalArticleSpec	Muß angegeben werden, wenn es dem Anwender ermöglicht werden soll, direkt (möglicherweise teilbestimmte) Endartikelnummern zur Artikelerzeugung eingeben zu können (d.h., der Artikel wird dann in der durch die Endartikelnummer codierten Konfiguration eingefügt)[a].

[a] Das Feature sollte aus Performance-Gründen und um potentielle Probleme bei der Auswertung der Endartikelnummer zu vermeiden, nur gesetzt werden, wenn die Endartikelnummer-Codierung die Verarbeitung einer eingegebenen (möglicherweise teilbestimmten) Endartikelnummern tatsächlich auch zulässt.

```
Bsp.: features = editableCatalog;showInitProgressDlg
```

[28] ***series_type*** **[O]**
– definiert spezifische Typen für das Paket

Der Schlüssel steuert typspezifische Behandlungen. Mehrere Typen werden durch ein Semikolon getrennt. Folgende Werte sind definiert:

Wert	Beschreibung
go_meta	Unterstützung von OFML–Metatypen 1.x

```
Bsp.: series_type = go_meta
```

[29] ***special_article_scheme*** **[O]**
– bestimmt eine spezifische Formatierung für Artikelnummern von Sonderausführungen

Der Wert kann aus beliebigen alphanumerischen Zeichen bestehen sowie aus den Ersetzungszeichen '?' und '*'. Ein '?' wird durch ein Zeichen der Original–Grundartikelnummer ersetzt. Die Ersetzung erfolgt der Reihe nach beginnend mit dem ersten Zeichen der Original–Grundartikelnummer. Das Zeichen '*' wird durch die komplette Original–Grundartikelnummer ersetzt. Standardmäßig wird 'SONDER *' (sprach-spezifisch) verwendet

```
Bsp.: special_article_scheme = CS??????
```

[30] ***~~progid_3d~~(obsolete)*** **[O]**
– bestimmt den Serienbestandteil für die 3D–Strukturinformation

Dieser Wert kann für Grafikexporte genutzt werden, welche Objektnamen in ihrer Struktur unterstützen (z.B. 3DS).

```
Bsp.: progid_3d = GOI
```

[31] ***~~layer_progid_2d~~(obsolete)*** **[O]**
– bestimmt den Serienbestandteil eines 2D-Layer-Namens

Dieser Wert kann für layerorientierte Grafikexporte genutzt werden (z.B. DXF).

```
Bsp.: layer_progid_2d = EGRGOIEX2D
```

[32] ***~~layer_progid_3d~~(obsolete)*** **[O]**
– bestimmt den Serienbestandteil eines 3D-Layer-Namens

Dieser Wert kann für layerorientierte Grafikexporte genutzt werden (z.B. DXF).

```
Bsp.: layer_progid_3d = EGRGOIEX3D
```

[33] ***~~block_progid_2d~~(obsolete)*** **[O]**
– bestimmt den Serienbestandteil eines 2D-Blocknamens

Dieser Wert kann für blockorientierte Grafikexporte genutzt werden (z.B. DXF).

```
Bsp.: block_progid_2d = EGRGOIEX2D
```

[34] ***~~block_progid_3d(obsolete)~~*** **[O]**
– bestimmt den Serienbestandteil eines 3D-Blocknamens

Dieser Wert kann für blockorientierte Grafikexporte genutzt werden (z.B. DXF).

```
Bsp.: block_progid_3d = EGRGOIEX3D
```

[35] ***~~geo_export_params(obsolete)~~*** **[O]**
– bestimmt spezielle Parameter für den Export geometrischer Daten

Mehrere Parameter werden durch ein Semikolon getrennt. Folgende Werte sind erlaubt:

Wert	Beschreibung
create_hashed_mat_names	Konvertiert Materialnamen zu Hash-Werten für das Mapping mit einer generierten MLI-Datei.
create_hashed_geo_names	Konvertiert Geometrienamen zu Hash-Werten.

Optionen für die Erzeugung der Artikelgeometrie in Anwendungen mit DWG-Unterstützung:

Wert	Beschreibung
use_proxy_geometries	Geometriedaten aus Beschreibung erzeugen. Langsamer und nicht immer robust, aber kleinere Datenmenge und bestmögliche Qualität. (Standard)
use_generated_meshes	Mitgelieferte generierte Geometriedaten benutzen. Schnell und robust, aber größere Datenmengen und nicht immer in bestmöglicher Qualität.

> Hinweis:
> Die Werte *use_proxy_geometries* und *use_generated_meshes* sind exklusiv, sie können nicht kombiniert werden.

```
Bsp.: geo_export_params = create_hashed_mat_names;create_hashed_geo_names
```

[36] ***gtin_id*** **[O]**
– Gibt die GTIN (Global Trade Item Number) des Pakets an (s. [gtin]).

Der Schlüssel kann 8, 12, 13 oder 14 Zeichen lang sein, entsprechend den verschiedenen Schemas GTIN–14, GTIN–13, GTIN–12 oder GTIN–8.

[37] ***cat_type*** **[O]**
– Beschreibt den Katalog-Typ des Pakets.

Der Schlüssel hat die Bildungsvorschrift:

```
cat_type=<Type>[:<Version>[.<MinorVersion>[.<SubMinorVersion>]]][/<PhysicalFormat>]
```

Folgende Typen sind definiert:

`XCF` Katalog im XCF–Format (Standard)

`OAS` Katalog im OAS–Format

`NULL` Paket ohne Katalog
(Die Anzeige möglicherweise vorhandener Katalogdaten wird damit explizit unterbunden.)

Die (optionale) Versionsangabe spezifiziert bei Katalog-Typen mit versionierten Datenformaten die verwendete Version.

Aktuell sind folgende physischen Formate definiert (optional):

- *CSV* (Standard)

```
Bsp. 1: cat_type = XCF:1.8/CSV
Bsp. 2: cat_type = NULL
```

[38] ***masked_catalogs*** **[O]**
– Bestimmt die Sichtbarkeit eines Katalogs in speziellen Anwendungen

Die Katalogdaten eines Paket können für bestimmte Anwendungen als „nicht anzuzeigen“ markiert werden. Die Anwendungen sind definiert durch die genormten Modul-Schlüssel und eine optionale Versionsnummer. So fern angegeben, erfordern die Teile der Versionsnummer ein exakte Übereinstimmung. Das Paket wird trotzdem registriert und ggf. geladen, aber nicht regulär angezeigt.

Der Schlüssel hat die Bildungsvorschrift:

```
masked_catalogs=<ModuleKey>[:<Version>[.<MinorVersion>[.<SubMinorVersion>]]][;...]
```

Version bezieht sich auf die Version des jeweiligen Moduls.

```
Bsp. 1: masked_catalogs = P-PL-X:6.3.0;P-XCAD:2
```

[39] ***persistency_form*** **[O]**
– bestimmt die persistente Repräsentation der Artikel des Pakets.

Folgende Werte sind erlaubt:

Werte	Beschreibung
SCENE	Artikelinstanzen müssen als vollständige OFML–Szene gespeichert werden.
STATECODES	Artikel können anhand ihrer Zustandscodes gespeichert werden, welche diese vollständig beschreiben.

Der Modus *STATECODES* führt zu einer kleineren Größe des gespeicherten Projektes, setzt aber voraus, dass alle Artikel des Pakets tatsächlich vollständig durch ihre Zustandscodes beschrieben sind.

Der Standardwert für *persistency_form* ist *STATECODES*.

```
Bsp.: persistency_form = STATECODES
```

Achtung:
Der Schlüssel wurde bislang nur vom pCon.planner ausgewertet. Für das Release des pCon.planner im Herbst 2023 wird dieser Schlüssel abgekündigt! Alle Artikel werden dann einheitlich im OBX–Format gespeichert.

[40] ***add_gfx_symbols.<format>*** **[O]**
– definiert die OFML-Bibliothek, welche zusätzliche Grafik-Symbole in dem angegebenen Format enthält.

Die zusätzlichen Grafik-Symbole können von einer Applikation zur Erzeugung einer Grafik für Artikel aus dem vorliegenden Paket in dem angegebenen Format verwendet werden.

Schlüssel-Syntax: `::($manufacturer)::($program)::`

Von der Applikation wird das Paket der angegebenen OFML-Bibliothek verwendet, welches in demselben Daten- bzw. Katalogprofil gelistet ist, aus dem auch das vorliegende Paket geladen wurde.

Folgende Formate werden aktuell unterstützt:

Format	Beschreibung
skp	Sketchup

5.4 Sprachabhängige Schlüssel

Die Registrierungsdateien können mehrere Sektionen für sprachabhängige Texte enthalten. Eine Sektion wird durch einen in eckigen Klammern eingeschlossenen Sektionsnamen eingeleitet. Die Namen der sprachabhängigen Sektionen entsprechen den zweistelligen ISO–Sprachen–Codes (ISO–639-1) (s. [iso639-1]). Die nachfolgend angegebenen Schlüssel dürfen mehrfach, in jeder Sektion, verwendet werden. Jeder Schlüssel überschreibt den Wert der evtl. schon in der zugehörigen Hersteller– bzw. Konzern–Registrierung hinterlegt ist.
Sprachunabhängige Schlüssel werden in den Sprachsektionen nicht gefunden, dürfen deshalb dort nicht verwendet werden.

[41] ***manufacturer_name*** **[O]**
– definiert die Bezeichnung des Herstellers bzw. Lieferanten

[42] ***program_name*** **[M]**
– definiert die Bezeichnung der Bibliothek

```
Bsp.: program_name = Büroelemente
```

[43] ***copyright*** **[O]**
– enthält einen Copyright–Vermerk

```
Bsp.: copyright = EasternGraphics 2004
```

[44] ***description*** **[O]**
– enthält eine Kurzbeschreibung der Bibliothek

```
Bsp.: description = Beispiele für die Anlage von OFML-- und Produktdaten
```

5.5 Beispiel

egr_office2_EGR_1.cfg:

```
# manufacturer information
manufacturer=egr
manufacturer_id=EG

# package information
program=office2
program_name=Office, V.2
program_id=OFFICE2

# catalog information
type=product
category=furniture
languages=en;de;es;fr;it;nl;pl;ru
distribution_region=EGR
version=1
release_version=1.0.19
release_date=2016-02-23
release_timestamp=20160225135032
cat_type=XCF

# product database
pd_format=OCD_4.0
productdb=::ofml::xoi::xOiNativeOCDProductDB40
productdb_path=egr/office2/EGR/1/db
oam_path=egr/office2/EGR/1/oam

# program information
proginfo=::ofml::xoi::xOiProgInfo
proginfodb_path=egr/office2/1

# relations and dependencies
depend=::ofml::go::1.14.9/ANY

# additional settings
series_type=go_meta
meta_type=::ofml::go::GoMetaType;::ofml::go::goGetMetaType()
geo_export_params=use_proxy_geometries

[de]
program_name=Büroelemente

[en]
program_name=Office elements
```

6 Daten– und Dateitypen

6.1 OFML– und Grafikdaten

OFML-Daten eines OFML–Paketes werden vertriebsgebiet-unabhängig unterhalb des Verzeichnisses

```
<data>/($manufacturer)/($program)/($version)
```

abgelegt. Dieses Verzeichnis definiert einen Namensraum. Daraus ergibt sich auch die Bildungsvorschrift für den Namen des OFML–Archives (`*.alb`). Folgende Dateiformate zählen zu den OFML– und Grafikdaten:

Verwendung	Dateiname	Endungen
OFML–Archive	$manufacturer_$program_$version	alb
OFML–Klassen*	$name	cls
String–Ressourcen*	$program_$lang-code	sr
3D–Geometrien*	$name	geo, 3ds, dwg
2D–Symbole*	$name	egms, fig
Texturen/Imagemaps*	$name	tga, jpg, bmp, png
OFML–Materialien*	$name	mat
EBase-Datenbank	$name	ebase
CSV-Datenbank	$name	csv
Material–Bibliotheken	$name	mli

Der Name der Datei bildet sich aus dem Namen des Objekts ($name) oder durch die Werte aus den Registrierungsdateien ($manufacturer), ($program).
Alle mit * gekennzeichneten Typen können und sollten in das OFML–Archiv aufgenommen werden. Für erweiterte Konzepte der Datenverteilung ist dies u.U. zwingend erforderlich.
CSV–Datenbanken sollten immer in das performantere EBase–Format umgewandelt und aus den Distributionsdaten entfernt werden[19].

6.2 OFML–Produktdaten

Produktdaten[20] werden unterhalb des Verzeichnisses

```
<data>/($manufacturer)/($program)/($region)/($version)/db
```

in einer Produktdatenbank abgelegt.

6.3 XCF–Katalog

Die Katalog–Informationen des OFML-Pakets im XCF–Format [xcf] werden unterhalb des Verzeichnisses

```
<data>/($manufacturer)/($program)/($region)/($version)/cat
```

abgelegt.

[19] EBase ist eine von EasternGraphics entwickelte Datenbank im Binärformat zum ausschließlich lesenden Zugriff. Eine EBase-Datenbank wird mittels einer Tabellenbeschreibungsdatei aus Textdateien mit fester oder variabler Feldlänge (z.B. CSV) generiert. Das Tool zur Erstellung einer EBase-Datenbank und die Beschreibungsdateien für die verschiedenen Arten von OFML-Daten können bei EasternGraphics angefordert werden.

[20] hier sind im engeren Sinn die kfm. Daten gemeint

6.4 Spezifische Ressourcen

Spezifische Ressourcen werden separat abgelegt in Unterverzeichnissen von:

```
<data>/($manufacturer)/($program)/($region)/($version)
```

Sie können HTML-Dokumente, Bild-, Video- und andere Daten beinhalten. Die Nutzung dieser Daten ist dem Anwendungssystem freigestellt. Nachfolgend werden die vordefinierten Ressource–Typen behandelt.

Ressourcen sonstiger, unten nicht explizit genannter Typen sind im Verzeichnis

```
<data>/($manufacturer)/($program)/($region)/($version)/etc
```

abzulegen.

6.4.1 Katalog–Bilder

Im Katalog referenzierte Bilddateien werden unterhalb des Verzeichnisses

```
<data>/($manufacturer)/($program)/($region)/($version)/image
```

abgelegt.

Die Bilddaten können auch für die Druckausgabe oder für die Referenzierung in HTML-Dokumenten benutzt werden.

Das Bildformat muss JPEG oder PNG sein[21].

Die Dateien in diesem Verzeichnis können auch in einem ZIP–Archiv `image.zip` hinterlegt sein (s.a. Abschn. 2.1).

6.4.2 Materialbilder

Als Materialbilder werden Bilddaten bezeichnet, die für die Visualisierung von Materialeigenschaften bzw. allgemein von Merkmalswerten verwendet werden[22].

Die Namen der Bilddateien (ohne Dateinamenserweiterung) werden anhand von Eigenschaften–Schlüssel und Merkmalswert mittels der OFML–Schnittstelle *Property* ermittelt.

Das verwendete Grafikdateiformat wird durch die Dateinamenserweiterung bestimmt. Für Materialbilder sind die Formate JPEG und PNG erlaubt, wobei eine Datei im Format PNG Vorrang hat[23].

Es werden zwei Arten der Ablage der Materialbilder unterstützt:

1. Diese ältere Art und Weise unterstützt die Ablage eines kleinen Icons in der Standardbildgröße von 50 x 18 Pixeln (Breite x Höhe) sowie eines größeren quadratischen Bildes in der Standardbildgröße von 50 x 50 Pixeln.

   Diese werden abgelegt im Verzeichnis:

   ```
   <data>/($manufacturer)/($program)/($region)/($version)/mat
   ```

[21] Zu den Bildformat-Konventionen siehe Abschn. 6.5.
[22] z.B. im Eigenschafteneditor
[23] Zu den allgemeinen Bildformat-Konventionen siehe Abschn. 6.5.

Der Dateiname für das quadratische Bild wird gebildet, indem an den eigentlichen Bildnamen noch ein Unterstrich angehängt wird.

Beispiel:

klein:
`<data>/($manufacturer)/($program)/($region)/($version)/mat/material.jpg`
quadratisch:
`<data>/($manufacturer)/($program)/($region)/($version)/mat/material_.jpg`

2. Um den Applikationen mehr Gestaltungsmöglichkeiten bei der Präsentation zu geben[24], können die Materialbilder auf diese neue Weise in folgenden 3 Größen hinterlegt werden:
   - kleines Icon für die kompakte Darstellung: 200 x 72 (Breite x Höhe in Pixel)
   - mittleres (quadratisches) Icon: 200 x 200 Pixel
   - großes Bild für die Material- bzw. Merkmalswertvorschau:
     - das Bild darf eine maximale Kantenlänge von 2000 Pixeln haben
     - die längste Kante sollte als Empfehlung mindestens 1000 Pixel lang sein
     - es gibt keine Vorgabe zum Seitenverhältnis

   Die Dateien für die verschiedenen Bildgrößen liegen in entsprechenden Unterverzeichnissen von

   `<data>/($manufacturer)/($program)/($region)/($version)/mat`

   `s` – kleine Icons
   `m` – mittlere Icons
   `l` – große Bilder

   Die Anlage des Unterverzeichnisses `l` ist optional. Zudem ist es dort nicht erforderlich, für alle Merkmalswerte ein Bild zu hinterlegen. Es muss nur ein Bild hinterlegt werden, wenn für den gegebenen Wert eine Material- bzw. Merkmalswertvorschau gewünscht ist.

   Beispiel:

   klein:
   `<data>/($manufacturer)/($program)/($region)/($version)/mat/s/material.jpg`
   mittel:
   `<data>/($manufacturer)/($program)/($region)/($version)/mat/m/material.jpg`
   groß:
   `<data>/($manufacturer)/($program)/($region)/($version)/mat/l/material.jpg`

Für eine optisch gute Darstellung gelten folgende allgemeine Empfehlungen:

- Materialbilder sollten keinen unnötigen Weißraum haben.
- Materialbilder dürfen keinen Rahmen/Rand oder ähnliches mitbringen.

Die Anzeige der Materialbilder durch die Applikationen ist optional und die Form der Darstellung hängt von der Anwendung ab. Die Applikationen sind dafür zuständig, die Bilder korrekt zu skalieren und darzustellen.

Existiert kein Unterverzeichnis `s` oder kein Unterverzeichnis `m`, werden weiterhin die bisherigen kleineren Bilder genutzt (gemäß der älteren Art der Datenanlage).
Ist ein Unterverzeichnis `s` *und* ein Unterverzeichnis `m` vorhanden, werden ausschließlich die dort hinterlegten Bilder von Anwendungen genutzt, die diese neue Art der Datenanlage unterstützen. D.h., es erfolgt kein Fallback auf die Bilder gemäß der älteren Art der Datenanlage, falls es für einzelne Merkmalswerte keine Bilder im Unterverzeichnis `s` bzw. Unterverzeichnis `m` gibt.

Sollen also alle Anwendungen mit der bestmöglichen Darstellungsqualität unterstützen werden, müssen die Materialbilder sowohl gemäß der älteren als auch der neuen Art der Datenanlage bereitgestellt werden.

[24] inkl. Unterstützung hochauflösender Ausgabegeräte

Die Dateien im Verzeichnis (inklusive der Unterverzeichnisse bei der 2. Ablagevariante)

```
<data>/($manufacturer)/($program)/($region)/($version)/mat
```

können auch in einem ZIP–Archiv `mat.zip` hinterlegt sein (s.a. Abschn. 2.1).

6.4.3 HTML–Dateien

In den Katalogdaten referenzierte HTML–Dateien können in den Verzeichnissen

```
<data>/($manufacturer)/($program)/($region)/($version)/html/($language)
<data>/($manufacturer)/($program)/($region)/($version)/html
```

abgelegt werden. Für jede unterstützte Sprache kann ein Unterverzeichnis mit dem jeweiligen zweistelligen ISO–Sprachen–Code (ISO-639-1) als Verzeichnisnamen angelegt werden. Eine referenzierte HTML–Datei wird immer zuerst in dem sprach-spezifischen Verzeichnis gemäß der aktuell in der Applikation eingestellten Katalogsprache gesucht, unabhängig davon, ob die zugehörige Resource im Katalog selber auch sprach-spezifisch angelegt ist, oder nicht.

HTML–Dateien können Bilddaten aus dem Bilddaten–Verzeichnis referenzieren.

6.4.4 Konfigurationen und Geometrien

Konfigurationen (.fml), OFML-Gruppen (.ogrp), Geometrien (.3ds, .dwg) sowie Container–Dateien (.pec), die in den Katalogdaten referenziert sind, werden abgelegt unter:

```
<data>/($manufacturer)/($program)/($region)/($version)/etc
```

6.4.5 Artikelspezifische Ansichten

Eine Tabelle mit artikelspezifischen Ansichten kann unter

```
<data>/($manufacturer)/($program)/($region)/($version)/etc/artsetups.csv
```

abgelegt werden. Diese Ansichten werden applikationsspezifisch ausgewertet. Das Format der Tabelle ist in [asv] beschrieben. Primär wird die Tabelle des Pakets verwendet, aus dessem Katalog der Artikel erzeugt wurde, alternativ die Tabelle aus dem im Katalog ggf. referenzierten Paket.

6.5 Bild–Format–Konventionen

Bilder im **JPEG**-Format müssen gemäß der Spezifikation für das „JPEG File Interchange Format (JFIF)“[25] hinterlegt werden:

- müssen sequenziell (nicht interlaced/progressive) strukturiert sein
- müssen Huffman-Kodierung (nicht arithmetische Kodierung) nutzen
- müssen das YCbCr-Farbmodell nutzen (nicht CMYK, kein Schwarz/Weiß)
- müssen 8 Bit pro Farbkanal nutzen (nicht mehr)

[25] http://www.w3.org/Graphics/JPEG/jfif3.pdf

Bilder im **PNG**-Format müssen gemäß der „PNG (Portable Network Graphics) Specification, Version 2.2“[26] hinterlegt werden:

- müssen sequenziell (nicht interlaced/progressive) strukturiert sein
- müssen das RGB-Farbmodell nutzen (3 Farbkanäle, nicht weniger, d.h., kein Schwarz/Weiß)
- müssen 8 Bit pro Farbkanal nutzen (nicht mehr)
- optional ist ein 8 Bit Alphakanal für transparente Bilder erlaubt
- animierte PNGs (APNG) werden nicht unterstützt

Eingebettete Metadaten (Thumbnails, EXIF, IPTC, ICC-Profile, Druckinformationen) werden bei beiden Formaten ignoriert (nicht ausgewertet).

6.6 Preisprofile

Preisprofile sind in [ppr] beschrieben und spezifiziert.

Versionierte Preisprofile, die im Zusammenhang mit Katalogen eines Herstellers bzw. einer Marke installiert werden (s. Schlüssel *ppr_version*, Abschn. 3.3.2), sind in folgendem Verzeichnis abzulegen:

```
<data>/($brand)/priceprofiles/($ppr_version)/
```

Dies bedeutet, dass für einen Katalog nur ein Preisprofil hinterlegt werden kann und dieses unter der Marke abzulegen ist, unter welcher der Katalog geführt wird (s. Schlüssel *brand*, Abschn. 3.3.2)[27].

Preisprofile im Rahmen eines herkömmlichen Hersteller–Datenstandes sind in folgendem Verzeichnis abzulegen:

```
<data>/($manufacturer)/priceprofiles/
```

Registriert ein Hersteller Daten sowohl über ein Katalogprofil als auch über ein Kompatibilitätsdatenprofil, so müssen entsprechend 2 Preisprofile in den oben genannten Pfaden bereitgestellt werden.

[26] http://libpng.org/pub/png/spec/1.2/png-1.2-pdg.html

[27] Falls der Katalog Daten mehrerer Hersteller beinhaltet, müssen die evtl. vorhandenen hersteller-spezifischen Preisprofile also zu einem gemeinsamen Preisprofil zusammengeführt werden.

7 Historie

3.6.0
- Präzisierung zur Verwendung von Unterverzeichnissen im Archiv `mat.zip`[2.1]
- Die Registrierung von Herstellern (Abschn. 4) wird nun vor der Registrierung von Paketen (Abschn. 5) beschrieben
- Präzisierungen in den Abschnitten 4.1 und 4.2
- Index-Postfixe zur Abbildung von mehrzeiligen Textfeldern sind in der Hersteller-Registrierung nur noch bei den Schlüsseln der Gruppe *address* erlaubt[4.2.2]
- Neue Schlüssel *external_catalog.url* und *external_catalog.name* in der Hersteller-Registrierung[4.2.2]
- Obsoleter Schlüssel *module_depends* entfernt[5.3]
- Schlüssel *supplier_id* und *distributor_name* entfernt[4.2.2,5.3,5.4]
- Schlüssel *concern_id*, *ppr_region_id* und *concern_name* aus der Paket-Registrierung entfernt[5.3,5.4]
- Der Schlüssel *manufacturer_name* ist in der Paket-Registrierung nun optional[5.4]

3.5.1
- Schlüssel *progid_3d*, *layer_progid_2d*, *layer_progid_3d*, *block_progid_2d*, *block_progid_3d* und *geo_export_params*[5.3] als obsolete markiert (das entsprechende Verhalten ist Bestandteil des Grafikexports der jeweiligen Applikation)

3.5.0
- Obsolete Datenprofil-Sektion *[libdef]* entfernt[3.2]
- Obsoleten Wert *ISO-8859-2* (ISO–Latin-2) für Schlüssel *encoding* entfernt[5.3]
- Der Schlüssel *release_timestamp*[5.3] ist nicht mehr obsolete (nun aber optional)
- Obsoleten Wert *attrsel* für Schlüssel *type* entfernt[5.3]
- Neuer Schlüssel *oap_program*[5.3]
- Aktualisierung der unterstützten Produktdatenformate im Schlüssel *pd_format*[5.3]
- Obsolete Schlüssel *visibility* und *insertion_mode* entfernt[5.3]
- Nicht unterstützte physische Formate *EBASE*, *DBF* und *SQLITE* beim Schlüssel *cat_type* entfernt[5.3]
- Standardwert für den Schlüssel *persistency_form* ist nun *STATECODES*[5.3]
- Aktualisierung des Abschnitts 5.5 – Beispiel(e)
- Das kleine Hersteller- und Konzernlogo wird nicht mehr unterstützt, dafür nun ein größeres Logo[4.2.3,4.3.3]
- Die Unterstützung von PNG für Hersteller- und Konzernlogo durch die Applikationen ist nun verpflichtend[4.2.3,4.3.3]
- Ungültigen Verweis auf `xcf.ebase` entfernt[6.3]
- Präzisierung zum Ablageort von nicht vordefinierten Ressourcen[6.4]
- Materialbilder können nun auch im Format PNG vorliegen[6.4.2]
- Es wird nun eine dritte Bildgröße für Materialbilder unterstützt[6.4.2]
- Neue Größen für kleine und mittelgroße Materialbilder[6.4.2]
- Neue Struktur für die Ablage von Materialbildern[6.4.2]
- Der Ablageort von Container–Dateien ist nun spezifiziert[6.4.4]
- Präzisierungen zu den Bild–Format–Konventionen[6.5]

3.4.1
- Präzisierung zum Standardwert für den Schlüssel *persistency_form*[5.3]

3.4.0
- Neuer Wert *maySetFinalArticleSpec* für den Schlüssel *features*[5.3]
- Veraltetes Feature *freeArticles* entfernt (Artikel können inzwischen immer in Sonderausführungen umgewandelt werden)[5.3]
- Präzisierungen zu den Schlüsseln *encoding*, *distribution_region*, *visibility* und *persistency_form*[5.3]
- Empfehlung zu verwendbaren Zeichen im Schlüssel *program_id*[5.3]
- Präzisierung zu im Katalog referenzierten HTML-Dateien[6.4.3]
- Diverse kleinere Korrekturen

3.3.0
- Geänderte Regelung zu Preisprofilen im Zusammenhang mit Katalogen[6.6]
- Spezifizierung der erlaubten Zeichen in den Schlüsseln *manufacturer*, *manufacturer_id* und *concern_id*[5.3]

3.2.0
- Neuer Schlüssel *add_gfx_symbols*[5.3]
- Korrigierte Verwendung der Begriffe „Bibliothek“ und „Paket“
- Veralteten Schlüssel *extpath* in Datenprofilen entfernt[3.2.3]
- Präzisierung zur Angabe von Paketen in Daten- und Katalogprofilen[3.2.3,3.3.4]
- Präzisierung zur Paket-Verwendung beim Schlüssel *catalogs*[5.3]
- Diverse kleinere Korrekturen

3.1.1
- Präzisierung zur Verwendung von Paketen mehrerer Hersteller in einem Katalogprofil[3.3.4]
- Präzisierungen zu den Schlüsseln *catalogs* and *pd_format*[5.3]

3.1.0
- Präzisierung zur Verwendung von Paketen mehrerer Hersteller in einem Katalogprofil[3.3.4]
- Präzisierung/Korrektur zur Prüfung von Abhängigkeiten zwischen Paketen (Schlüssel *depend*[5.3])
- Produktdatenformat EPL wird nicht mehr unterstützt (Schlüssel *pd_format*[5.3])
- Logos können optional auch im PNG-Format vorliegen[3.1.3,4.2.3,4.3.3]
- Katalogbilddateien können auch im PNG-Format vorliegen[6.4.2]
- Neuer Abschnitt zu Bild–Format–Konventionen[6.5]

3.0.1
- Korrektur beim Schlüssel *depend* bezüglich der Auswertung der Versionsangabe

3.0.0
- Einführung der Katalogprofile[3]
- Präzisierungen zur Hersteller-Registrierung[4.2.1,4.2.2]
- Präzisierung hinsichtlich der verwendbaren Zeichen im Schlüssel *program*[5.3]
- Präzisierungen zu den Schlüsseln *release_version*, *depend* und *persistency_form*[5.3]
- Schlüssel *release_timestamp*[5.3] als obsolete markiert
- Bibliothekstyp *attrsel* als veraltet markiert
- Schlüssel *extensions* entfernt (bei Bedarf *depend* verwenden!)
- Veraltete Schlüssel *gf_version* und *xcf_format* entfernt
- Präzisierung zum Ablageort von Preisprofilen[6.6]
- Reorganisation der Dokumentstruktur
- Kleinere Korrekturen

Für ältere Historie siehe ältere Versionen.