CityGML 3 Writer 入門 Version 1
製品 ID: FMWDK-CITYGML3-WRITER-INTRODUCTION
ダウンロードファイル: fmwdk-citygml3-writer-introduction_v_1_0_0.zip

FME 2026.1 で追加された OGC CityGML 3 Writer（注 1）を使い、複数の異なる詳細度、ジオメトリ表現の CityGML 3.0（GML Encoding）形式建築物データを作成するワークスペース例のセットを提供します。

Version 1 では、外部 LOD0・LOD1、内部 LOD0 の建築物データを作成するワークスペース例を提供します。
Version 2 以降で、LOD2、LOD3、建築物部材（BuildingConstructiveElement）表現、点群（PointCloud）表現の建築物データを作成するワークスペース例を追加する予定です（注 2）。

注 1: FME 2026.1.x の CityGML 3 Writer は Tech Preview の位置づけであり、未完成の機能や未解決の不具合があります。今後のバージョンで完全版がリリースされるまでの間は、CityGML 3 Writer が出力する CityGML データセットには不備がある可能性があることについて、ご承知おきください。

注 2: CityGML 3.0 では LOD（Level of Detail）の区分が従来（CityGML 2.0）の 5 段階から 4 段階（LOD0～LOD3）に変更されるとともに、各 LOD のジオメトリ表現の方法や組み合わせについての自由度が大きくなりました。本製品における LOD の定義は簡明さを重視しつつ、応用のしやすさも考慮した独自のものであって、国土交通省 3D 都市モデル整備事業（PLATEAU）などの特定の事業分野における LOD の定義を準用したものではないことにご留意ください。

本製品が提供するワークスペース例で使用する入力データ（Shapefile、FBX 等）は、カールスルーエ工科大学自動化・応用コンピューター科学研究所（Institute for Automation and Applied Computer Science (IAI) / Karlsruhe Institute of Technology (KIT)）が作成し、ウェブ上で公開している CityGML 3.0 データ例（FZK Haus CityGML 30）に基いて作成しました。

稼働環境: FME 2026.1 以降
※ワークスペース例の作成、動作確認は FME 2026.1.2, Windows 11 で行いました。

作成者: 飯嶋孝史

提供サイトドメイン: data-xformer.com

更新履歴:

年月日	バージョン	更新内容
2026-06-15	1.0.0	外部 LOD0, LOD1, 内部 LOD0

本書の構成

※以下の章は Version 2 以降で追加予定。

1. 外部 LOD0, LOD1 および内部 LOD0

1.1 LOD0, LOD1 の定義

(1) LOD0, LOD1 の定義

本製品では、建築物外部の LOD0, LOD1、および、建築物内部の LOD0 を表 1.1 のように定義します。

表 1.1 建築物外部 LOD0, LOD1, 建築物内部 LOD0 の定義

LOD	本製品での定義	GML 幾何型
外部 LOD0	建築物外形のフットプリントを表す多角形（z=0）	gml:MultiSurface
外部 LOD1	建築物外形のフットプリントを表す多角形（z=接地面標高）を鉛直上向きに押し出すことによって形成される柱状の立体（押出距離=建築物の高さ）	gml:Solid
内部 LOD0	部屋の床面を表す多角形（z=0）	gml:MultiSurface

(2) Storey の取り扱い

CityGML 3.0 の仕様では、建築物の外部、内部ともに、Storey（階）の単位でフィーチャーをグループ化できることになっていますが（必須ではなく任意）、本製品では統一的に、

- 外部のフィーチャーは Storey によるグループ化をしない
- 内部のフィーチャーは Storey によるグループ化をする

こととします。

変換結果の CityGML 文書では、建築物外形を表現するフィーチャーはどの Storey の階層にも所属せず、建築物内部の部屋を表現するフィーチャーは必ずどれかひとつの Storey の階層内にあり、Storey は Building の階層に所属するという階層構造を持つことになります。

```
Building（建築物）
  建築物外形
  Storey（階）1
    部屋 1-1
    部屋 1-2
    ,,,
  Storey（階）2
  ...
```

1.2 入力データセット

本章のワークスペース例（外部 LOD0 のみ、外部 LOD0-LOD1、内部 LOD0 のみ、外部 LOD0-LOD1＋内部 LOD0 の 4 パターン）では、以下に掲げるデータのひとつ以上を入力データセットとして使用します。

- building_footprint.shp: Shapefile 形式，建築物外形フットプリントポリゴン，平面直角座標 9 系
- room_floor.shp: Shapefile 形式，部屋床面ポリゴン，平面直角座標 9 系
- relations.csv: CSV 形式，フィーチャー間の関係定義 ※「1.5 ワークスペース例 1-03: 内部 LOD0」参照

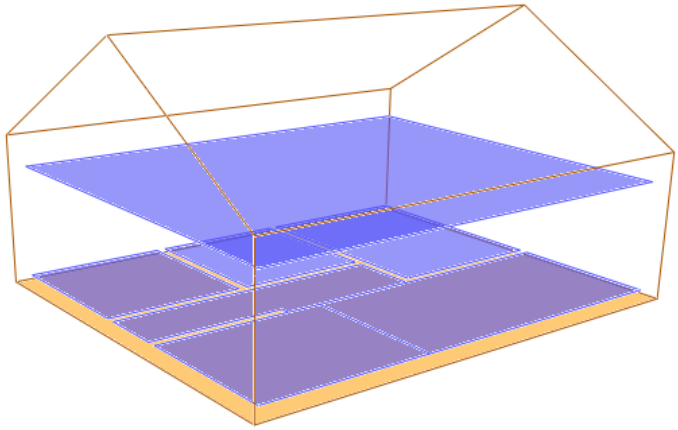

図 1.1 入力データイメージ: 建築物外形フットプリントおよび各部屋の床面
※図では立体的に表現していますが、実際の Shapefile データは 2D です。

Shapefile 形式のポリゴンデータの属性の構成を表 1.2 に示します。

表 1.2 建築物外形フットプリント / 各部屋の床面 Shapefile 形式ポリゴンデータの属性

属性名	説明
name	建築物 / 部屋の名称
feature_id	入力データセット内でのフィーチャー識別子: {名前空間略称}_{フィーチャータイプ名}_{連番}（注）
zmin	建築物底面 / 部屋床面の標高
zmax	建築物屋根面 / 部屋天井面の標高（最高点）

注: 変換結果の CityGML 文書において複数のフィーチャータイプを記述する場合に、入力データから読み込んだフィーチャーと出力先の CityGML 文書におけるフィーチャータイプを対応づける仕組みが必要です。さまざまな方法が考えられますが、本製品では、入力フィーチャーの feature_id 属性に、CityGML 文書における名前空間略称とフィーチャータイプ名を含める方針としました。

1.3 ワークスペース例 1-01: 外部 LOD0

ワークスペース	1-01_exterior-lod0.fmw
入力データ	source/1_lod0-1/building_footprint.shp（建築物外形のフットプリント）
変換結果	output/1-01_exterior-lod0.gml

注: ワークスペース（*.fmw ファイル）、入力データ（source フォルダー以下）、変換結果（output フォルダー以下）は、ダウンロードファイル内の examples フォルダー内に収録されています。以下同。

このワークスペース例は、以下のような直線的な（分岐、合流、結合がない）データフローによって、建築物外形のフットプリント（Shapefile 形式ポリゴンデータ）を建築物モデル（外部 LOD0 幾何オブジェクトを持つ）に変換するものです。

1) Shapefile Reader [building_footprint] : フットプリントフィーチャー（ポリゴンジオメトリ）読込
2) AttributeCreator : ★gml_id 属性の設定
3) Orientor - 3DForcer - FaceReplacer - Aggregator : 妥当なジオメトリ（MultiSurface）に変換 (注 1, 2)
4) GeometryPropertySetter : ★幾何オブジェクトの関連役割名の設定
5) AttributeManager : ★フィーチャーの関連役割名の設定、★主題属性の設定（例: gml:name）
6) CityGML 3 Writer [bldg_Building] : ★CityGML 3 Writer による GML 文書作成・出力

注 1: Orientor では、ポリゴンの面の向きを Left hand rule に修正しています。面を表側から見て、その境界線を頂点順に移動したときの左側が面の内側になることを Left hand rule と言い、OGC では、これを標準としています。ところが、一部のデータ形式、アプリケーションでは、Right hand rule を採用しているものがあります。Shapefile はそのひとつで、Right hand rule のまま CityGML の LOD0 幾何オブジェクトに変換すると、裏返しの面になってしまいます。Left hand rule、Right hand rule のどちらであるかは、FME Workbench | Data Preview または FME Data Inspector の Record Information ウィンドウで確認できます。

注 2: 今後の FME のバージョンアップに伴う CityGML 3 Writer の改良により、サポートするジオメトリタイプの範囲が広くなる可能性があります。それが実現すると、この処理ブロックのうち FaceReplacer, Aggregator は不要になります。

上記フローのうち★印を付した事項は、CityGML 3 Writer によって CityGML 3.0 形式のデータを作成するワークスペースでは必須です。最初のワークスペース例なので、以下、それらについて詳述します。FME Workbench でワークスペース例の内容を参照するとともに、テキストエディタで変換結果 CityGML 文書を確認しながら、しっかり理解してください。

(1) gml:id 属性の設定（gml_id 属性の作成）

CityGML 文書におけるフィーチャー要素（この例では Building）には、gml:id 属性（値はユニークでなければならない）を設定する必要があります。

CityGML 3 Writer で CityGML 文書を作成する場合、出力するフィーチャーに"gml_id"という名前の属性を付与すると、その値が、出力先の文書におけるそのフィーチャーに対応する要素の gml:id 属性の値に反映されます。

このワークスペース例では、AttributeCreator によってフィーチャーに"gml_id"属性を追加し、FME の@UUID 関数によって生成した UUID（Universally Unique Identifier）に"bldg_"という接頭辞をつけた値を設定しました。

一意性は UUID によって担保されますが、UUID は先頭が数字になることがあります。XML スキーマ上、gml:id 属性は xs:ID 型と定義されており、そのデータ型の文字列は数字から始めることはできないため、英文字から始まる接頭辞をつけることによって xs:ID 型の妥当な文字列となるようにしました。

(2) 幾何オブジェクトの関連役割名を設定

CityGML 文書において LOD 別に記述される幾何オブジェクト（gml:MultiSurface, gml:Solid など）は、次のような名前の親要素（要素名＝関連役割名）を介して、フィーチャーと関連づけられます。

lod{N}{幾何オブジェクト型名等}（{N}: 0-3）例: lod0MultiSurface, lod1Solid など

次の例では、LOD0 の幾何オブジェクト gml:MultiSurface が lod0MultiSurface によって Building フィーチャーに関連付けられています。言い換えれば、建築物の外部 LOD0 の形状が、gml:MultiSurface で記述されているということになります。

```
<bldg:Building>
    <gml:name>KIT FZK-Haus</gml:name>
    <core:lod0MultiSurface>
        <gml:MultiSurface>
            <gml:surfaceMember>
                <gml:Polygon>
                (以下略)
```

CityGML 3 Writer で CityGML 文書を作成する場合、ジオメトリのルートノード（注）に Geometry Label として設定した文字列が、幾何オブジェクトの親要素名として出力先の CityGML 文書に反映される仕組みとなっています。

注: FME の内部では、一般に、ジオメトリは XML のような階層構造をもったデータとして取り扱われています。例えば MultiSurface 型のジオメトリは、[ルート] MultiSurface / [中間] Part (Face 等) / [中間] Area (Polygon 等) / [リーフ] Boundary (Line 等) という階層構造をもっていて、各階層のジオメトリには、必要に応じてプロパティ（Geometry Label [0..1], 複数の Trait [0..*]）を設定することができます。ジオメトリの構造やプロパティは、FME Workbench | Data Preview または FME Data Inspector の Record Information ウィンドウで確認できます。

Geometry Label の設定は、次のパラメーター設定による GeometryPropertySetter によって行います。

- Property to Set: Geometry Label
- Overwrite Existing Properties: Yes
- Geometry Label: lod0MultiSurface　※パラメーター設定フィールドに直接キー入力

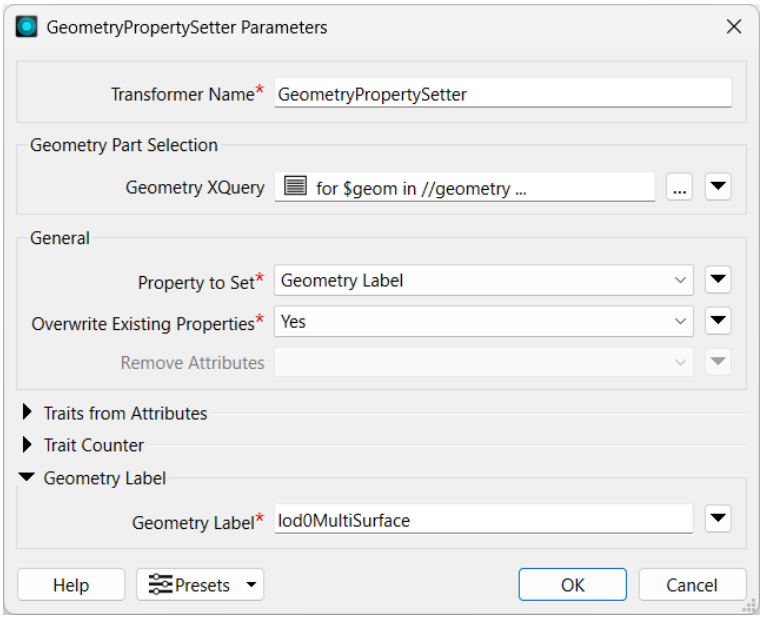

図 1.2 GeometryPropertySetter パラメーター設定（Geometry Label の設定）

この例の場合は、GeometryPropertySetter | Geometry XQuery パラメーターのデフォルトの設定によってジオメトリ内のすべてのノードに Geometry Label を設定しても差し支えありませんが、同パラメーターの設定によって明示的にルートノードだけに設定することもできます。

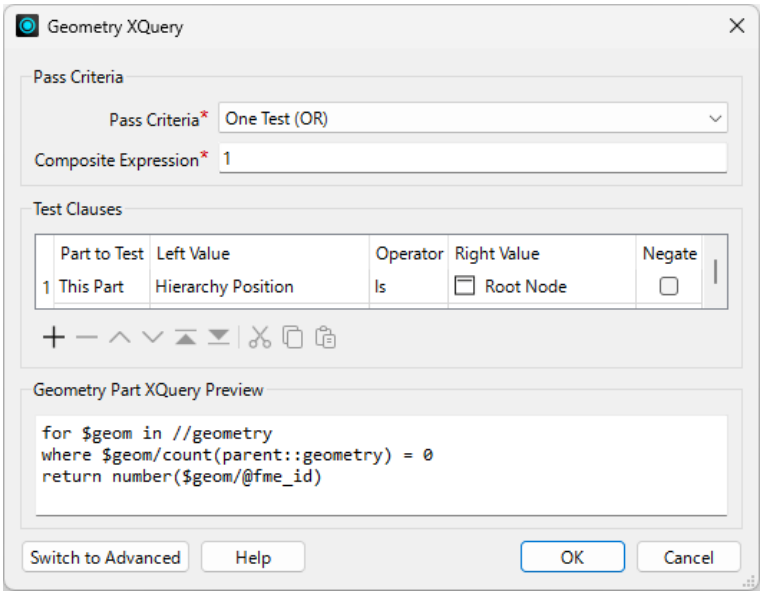

図 1.3 GeometryPropertySetter | Geometry XQuery パラメーター設定（ルートノード選択）

ジオメトリに設定されているプロパティは、FME Workbench | Data Preview または FME Data Inspector の Record Information ウィンドウで確認できます。

(3) フィーチャーの関連役割名の設定（gml_parent_property 属性の作成）

CityGML 文書では、フィーチャー要素の親要素名によって、その上位のフィーチャーとの関連役割が明示されます。次

の例では、Building（建築物）が cityObjectMember（都市オブジェクトメンバー）役割を持ったフィーチャーとして、上位の CityModel（CityGML 文書のルート要素）に関連付けられています。

```
<core:CityModel>
    <core:cityObjectMember>
        <bldg:Building>
            <gml:name>KIT FZK-Haus</gml:name>
            (以下略)
```

CityGML 3 Writer で CityGML 文書を作成する場合、出力するフィーチャーに"gml_parent_property"という名前の属性を付与し、その値として適切な関連役割名を格納することにより、それが出力先 CityGML 文書に反映されます。ワークスペース例では、AttributeManager で"gml_parent_property"属性を設定しました。

注: cityObjectMember（CityModel 直下のフィーチャーの関連役割名）は"gml_parent_property"属性を介した設定を省略できるのですが、本製品のワークスペース例では、出力先 CityGML 文書の構成に関係するワークスペース設計の原則を示すという意味で、cityObjectMember についても明示的に"gml_parent_property"属性による設定をしました。

(4) 主題属性（例: gml:name）の設定

CityGML のフィーチャーは、その下位の要素によって主題属性を持つことができます。どのような主題属性要素を持てるかは XML スキーマで規定されており、gml:name（名称）はそのひとつです。

CityGML 3 Writer で CityGML 文書を作成する場合、出力するフィーチャーに、設定したい主題属性要素名と語幹部が等しい名前の属性を付与することによって、その値を出力先 CityGML 文書における当該主題属性要素の値に反映させることができます。

主題属性要素に対応する属性について、次の 2 点に注意してください。

- 名前空間プレフィクスの有無: CityGML 3 Writer | Add XML Namespace Prefix to パラメーターの設定（後述）により、主題属性要素に対応する属性名に名前空間プレフィクスをつける必要があるかどうかが決まります。
- 主題属性要素の多重度: XML スキーマ上、複数回記述し得る主題属性要素の場合、それに対応する属性はリスト属性とし、反映させる値をリストの要素として格納する必要があります。スキーマ上、最大でもひとつしか記述できない主題属性要素の場合は、（リストではない）通常の属性とします。

ワークスペース例では、gml:name 要素の値を設定するために、AttributeManager によって、入力データが持っていた"name"属性の名前を"name{0}"（リスト属性"name{}"の最初の要素）に変更することによって、gml:name（1 個）を設定しました。

これは、後述するように、CityGML 3 Writer | Add XML Namespace Prefix to パラメーターで属性名には名前空間プレフィクスをつけない設定をしたこと、および、CityGML 3.0（が引用する GML 3.2）の XML スキーマでは、gml:name の多重度が [0..*]（0 個以上、上限数なし）と定義されていることによります。

入力データにおいてひとつしか存在しない属性であっても、XML スキーマ上は複数記述し得る主題属性に設定する場合には、リスト属性の要素として CityGML 3 Writer に渡さなければならないことに注意してください。

(5) OGC CityGML 3 Writer 追加の手順

ワークスペース例に CityGML 3 Writer およびライターフィーチャータイプ bldg_Building を追加した手順とパラメーター設定について説明します。

1) Add Writer

Workbench メニュー: Build > Writers > Add Writer の選択、または、Add Writer ツールボタンのクリックによって Add Writer 画面を開き、Format として OGC CityGML 3 を選択します。

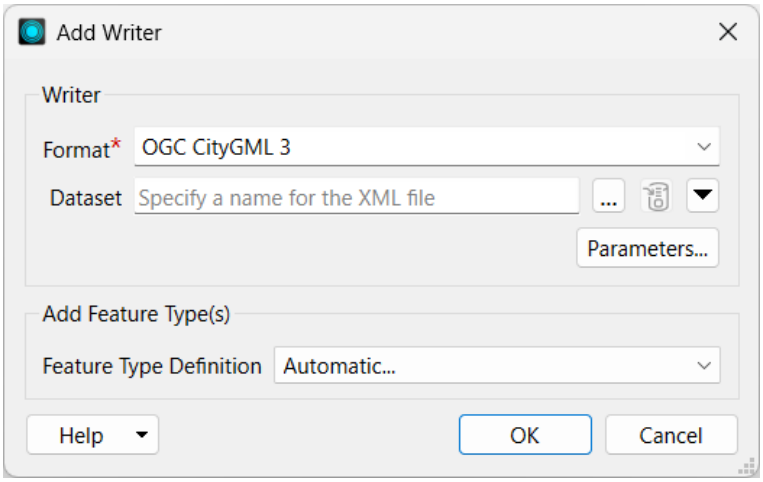

図 1.4 Add Writer 画面: Format として OGC CityGML 3 を選択

2) ライターパラメーターの設定

Add Writer 画面の [Parameters] ボタンをクリックしてパラメーター設定画面を開き、ライターのパラメーターを設定します。ワークスペース例では、次のように設定しました（その他のパラメーターはデフォルトの設定のまま）。

- Add XML Namespace Prefix to: Feature Types
- Pretty Print: Yes
- GML srsName: http://www.opengis.net/def/crs/EPSG/0/11327
- GML SRS Axis Order: 2,1,3

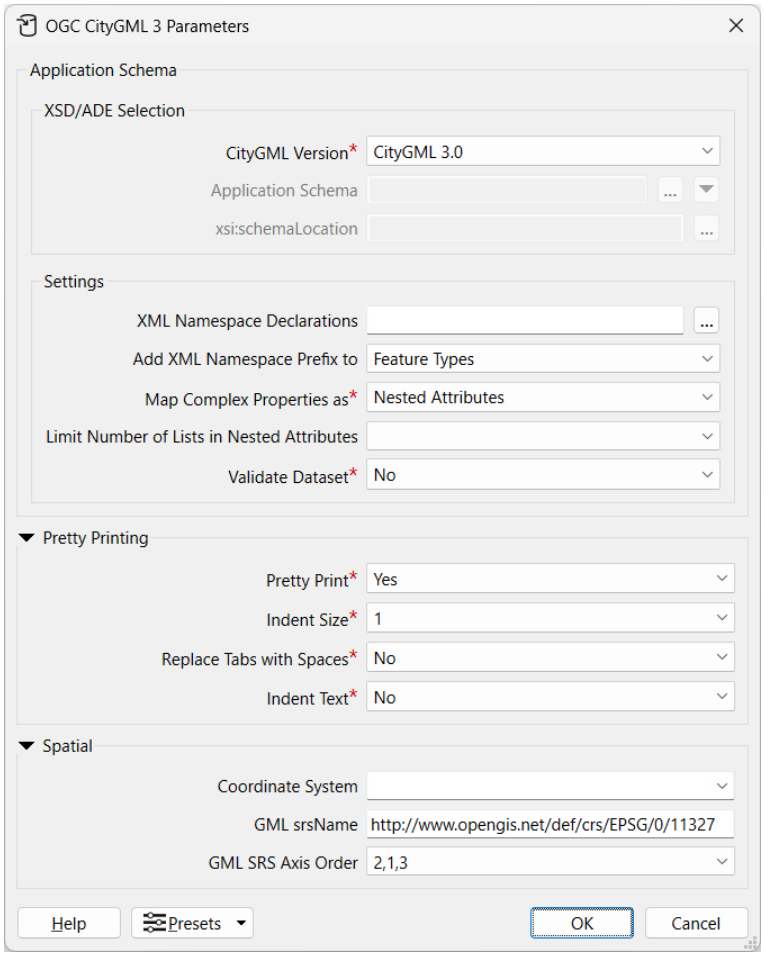

図 1.5 OGC CityGML 3 Writer パラメーター

Add XML Namespace Prefix to パラメーターでは、ワークスペース内で取り扱う CityGML のフィーチャータイプ名や主題属性に対応させる属性名にアンダースコア（下線）をはさんで XML 名前空間プレフィクスをつけるかどうかをコントロールします。

ワークスペース例では、このパラメーターで"Feature Types"（フィーチャータイプ名に XML 名前空間プレフィクスをつける。属性名にはつけない）を選択しました。これにより、Building フィーチャーを出力するためのライターフィーチャータイプ名は"bldg_Building"（プレフィクスあり。後述）、gml:name 要素に対応する属性名は"name{0}"（プレフィクスなし。前述）とすることになります。

あくまでも FME の処理において取り扱うフィーチャータイプ名、属性名にプレフィクスをつけるかどうかをコントロールするものであって、このパラメーターの設定内容とは関係なく、出力先の CityGML 文書における要素名は常に、コロンをはさんで適切な名前空間修飾子がついたものとなります（例: bldg:Building, gml:name）。

Pretty Print パラメーターでは、出力先の CityGML 文書において、各要素の行頭に、階層の深さに応じたサイズのインデント（空白）を挿入するかどうかを設定します。

ワークスペース例では、変換結果をテキストエディタで開いてデータ構造を確認しやすくするため、"Yes"を設定しました。インデントの有無によってデータの内容が変わることはなく、この設定は必須ではありません。

GML srsName パラメーターでは、出力先の CityGML 文書において srsName 属性の値として記述する座標参照系識別子を設定します。

ワークスペース例では、このパラメーターに"http://www.opengis.net/def/crs/EPSG/0/11327"（JGD2024 平面直角座標 9 系+標高を示す EPSG コードの URI 表記）を設定しました。入力データ（Shapefile 形式）に設定されている座標参照系は平面直角座標 9 系なので、投影法は変換は不要です。

GML SRS Axis Order パラメーターでは、出力先の CityGML 文書内の gml:posList 要素等における座標値の記述順を指定します。1 軸:東向き正の座標軸、2 軸:北向き正の座標軸、3 軸:鉛直上向き正の座標軸を示します。

ワークスペース例では、"2,1,3"、すなわち、平面直角座標系の X 軸（北向き）、Y 軸（東向き）、標高の順で記述するように設定しました。

CityGML や GML の仕様では 1 軸、2 軸の記述順は規定されておらず、座標参照系の登録先（レジストリ）において登録されている座標軸順にしたがうこととされています。srsName 属性で指定した座標参照系の識別子は EPSG コードを使っているので、EPSG に登録されている平面直角座標系 9 系の座標軸順（northing, easting）に準じて"2,1,3"としました。

なお、これらのパラメーターは、ライターをワークスペースに追加した後、FME Workbench | Navigator ウィンドウ上の操作で設定、変更することもできます。

3) フィーチャータイプ定義方法の選択

パラメーター設定画面を [OK] ボタンで閉じて、Add Writer 画面に戻ります。

ワークスペース例では、Feature Type Definition パラメーターで"Manual"を選択しました。

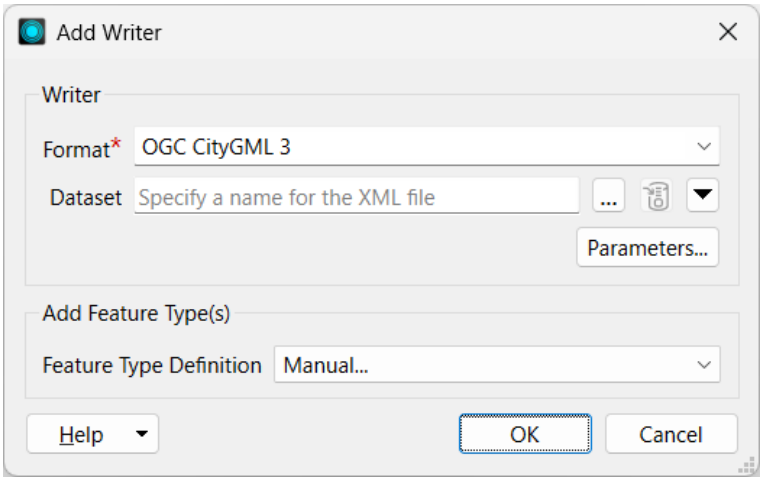

図 1.6 Add Writer 画面: Feature Type Definition パラメーターで"Manual"を選択

"Manual"の状態でライターを追加した場合、キャンバス上に追加されるフィーチャータイプの User Attributes は、ユーザー（ワークスペース作成者）が自由に設定、編集できます。しかし、CityGML 3 Writer の場合は、出力先の CityGML 文書に書き込める主題属性は XML スキーマで規定されており、User Attributes に何も設定されていなくても、CityGML 3 Writer は XML スキーマを参照して自動的にそれらに対応する属性名を認識するので、通常は、ユー

ザーがフィーチャータイプの User Attributes で属性名等を設定したり編集したりする必要はありません。ここでの"Manual"の選択は、User Attributes では何の設定も編集もしない、ということを意図しています。

なお、このパラメーターで"Import from Dataset"を選択すると、[OK] ボタンで閉じたときに FME は CityGML 3.0 の XML スキーマ（および、ADE を使う場合はその XML スキーマ）を参照して選択可能なフィーチャータイプ名のリストを表示し、ユーザーが必要なフィーチャータイプを選択してワークスペースに追加したときに、各フィーチャータイプでは、XML スキーマ上記述し得るすべての主題属性に基づいて、User Attributes の属性テーブルが自動的に構成されます。出力するフィーチャータイプが多数ある場合や主題属性に対応する属性名をライターフィーチャータイプのインターフェースで確認したい場合など、有用なこともあります。

4) ライターの追加

Add Writer 画面を [OK] ボタンで閉じると、キャンバス上に NewFeatureType というフィーチャータイプが現れ、そのプロパティ設定画面が開きます。

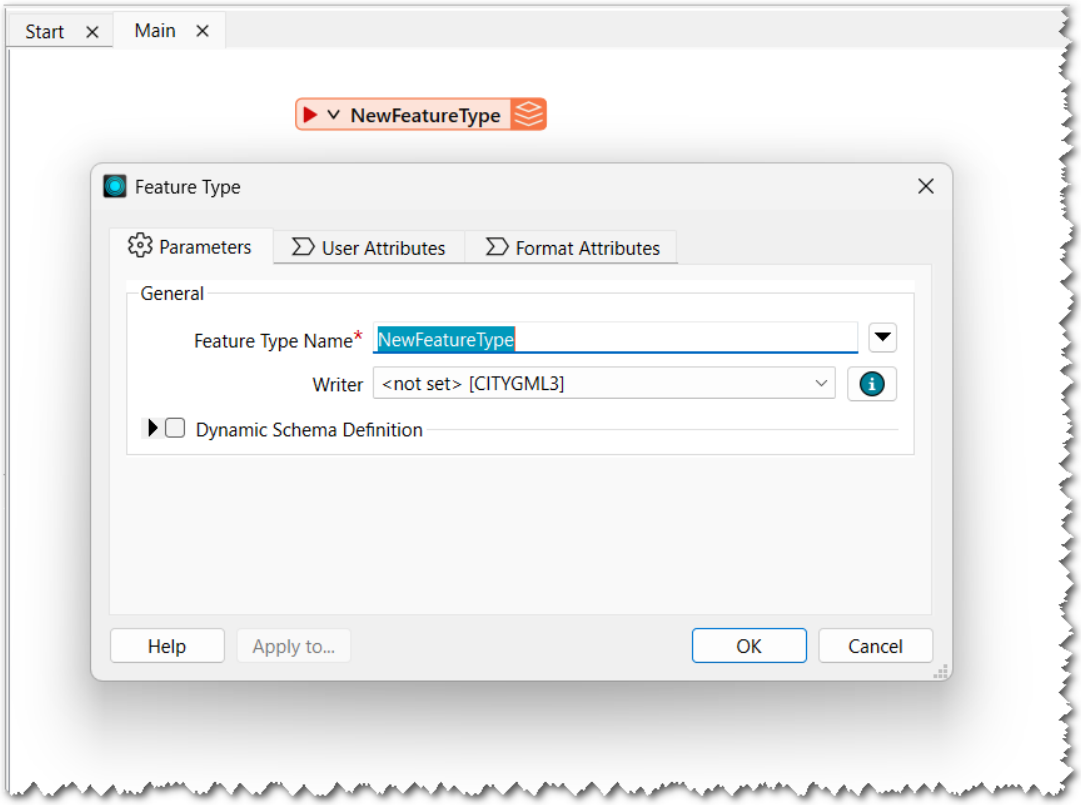

図 1.7 ライターフィーチャータイプ（フィーチャータイプ名変更前）

ここで、Feature Type Name パラメーターを"bldg_Building"（名前空間プレフィクスつきの CityGML 3.0 建築物フィーチャータイプ名）に変更してから [OK] ボタンで閉じると、ライターがワークスペースに追加されるとともに、フィーチャータイプ名が"bldg_Building"に変更されます。Feature Type Name を変更せずに一旦閉じてライターを追加した後、再度プロパティ設定画面を開いて変更することもできます。

これで CityGML 3 Writer による CityGML 3.0 データセット（Building フィーチャー）の出力準備は完了です。Navigator ウィンドウ上に [CITYGML3] ライターのノードが追加されていることも確認してください。

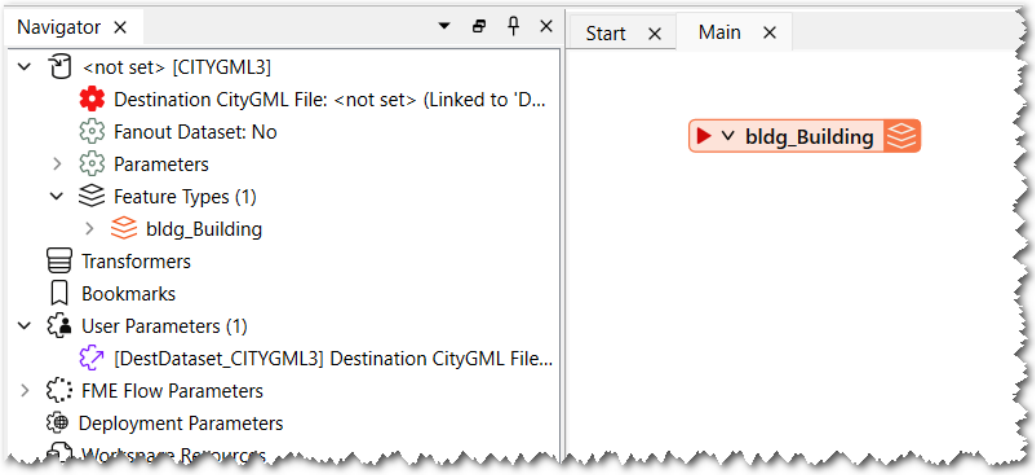

図 1.8 Navigator 上の [CITYGML3] ライターとキャンバス上の bldg_Building フィーチャータイプ

本節で説明した内容のうち、CityGML 3 Writer によって CityGML 3.0 形式のデータを作成するワークスペース作成において知っておくべき共通事項について、以下のとおりまとめます。

- フィーチャー要素の gml:id 属性の値は、gml_id 属性によって設定する。
- 幾何オブジェクトの関連役割名は、ジオメトリのルートノードに Geometry Label として設定する。
- フィーチャーの関連役割名は、gml_parent_property 属性によって設定する。
- 主題属性要素の値は、対応する属性によって設定する。ただし、XML スキーマ上 2 以上記述し得る主題属性要素の値は、リスト属性によって設定する。
- ライターフィーチャータイプ名、属性名に名前空間プレフィクスをつけるかどうかは、CityGML 3 Writer の Add XML Namespace Prefix to パラメーターによって選択する（本製品が提供するワークスペース例では、フィーチャータイプ名に名前空間プレフィクスをつけ、属性名にはつけない設定で統一した）。

また、本節では OGC CityGML 3 Writer をワークスペースに追加する手順についても説明しました。
次節以降では、これらについての詳細説明は繰り返しませんので、必要に応じて本節に戻って見直してください。

1.4 ワークスペース例 1-02: 外部 LOD0, LOD1

ワークスペース	1-02_exterior-lod0-1.fmw
入力データ	source/1_lod0-1/building_footprint.shp（建築物外形のフットプリント）
変換結果	output/1-02_exterior-lod0-1.gml

このワークスペース例は、前節のワークスペース例をベースとして、外部 LOD1（建物外形フットプリントを鉛直上向きに押し出して形成される柱状の立体）のジオメトリを作成する処理を追加し、建築物外部の LOD0 幾何オブジェクトと LOD1 幾何オブジェクトを併せ持つマルチスケール（multi-scale）モデルの CityGML 建築物フィーチャーを作成するように拡張したものです。変換結果の CityGML 文書において、1 棟の建築物フィーチャーは、概略次のような構造の XML テキストで記述されます。

```
<bldg:Building>
    <gml:name>KIT FZK-Haus</gml:name>
    <core:lod0MultiSurface> <!-- LOD0 -->
        <gml:MultiSurface>
        (略)
        </gml:MultiSurface>
    </core:lod0MultiSurface>
    <core:lod1Solid> <!-- LOD0 -->
        <gml:Solid>
        (略)
        </gml:Solid>
    </core:lod1Solid>
</bldg:Building>
```

以下、前節のワークスペースに追加した事項について説明します。

(1) フットプリントポリゴンから LOD1 ジオメトリ（立体）への変換

前節のワークスペースにおいて入力フィーチャーに"gml_id"属性を設定した後、フットプリントポリゴンを立体（ソリッド）に変換するための次のようなデータフローを追加しました。各トランスフォーマーのパラメーター設定については、ワークスペースを FME Workbench で開いて確認してください。

- 3DForcer: 2D ポリゴンを z = zmin（建築物底面の標高）の 3D ポリゴンに変換
- Extruder: 鉛直上向きに、建築物の高さ（zmax - zmin）だけ押し出した立体に変換（注）

注: 建築物屋根面の 3D 形状が航空レーザー測量で得られた点群データで与えられた場合、LOD1 立体の上面の標高の求め方にはいくつかの考え方がありますが、この例では単純に、入力データで与えられる zmax 属性の値（最高点の標高）としました。

なお、Extruder によるジオメトリの変換結果は、底面の形状と押出方向・距離を表す 3D ベクターによって定義され

る Extrusion というタイプの立体ですが、CityGML 3 ライターで出力する際に、立体を取り囲む各境界面によって構成する構造の gml:Solid に変換されます。

(2) LOD1 幾何オブジェクトの関連役割名を Geometry Label として設定

GeometryPropertySetter により、ジオメトリ（ソリッド）のルートノードの Geometry Label に、出力先幾何オブジェクトの関連役割名"lod1Solid"を設定しました。

(3) LOD0 ジオメトリと LOD1 ジオメトリの集約

複数の幾何オブジェクトを同時に持つマルチスケールモデルを CityGML 3 Writer で出力するには、それを構成する各幾何オブジェクトに対応するジオメトリにそれぞれの関連役割名を Geometry Label として設定した後、Aggregator によってそれらを集約して単一の集約ジオメトリ（Aggregate）にしました。集約後のジオメトリの構造（Aggregate を構成するジオメトリ、それらの Geometry Label）は、FME Workbench | Data Preview または FME Data Inspector の Record Information で確認できます。

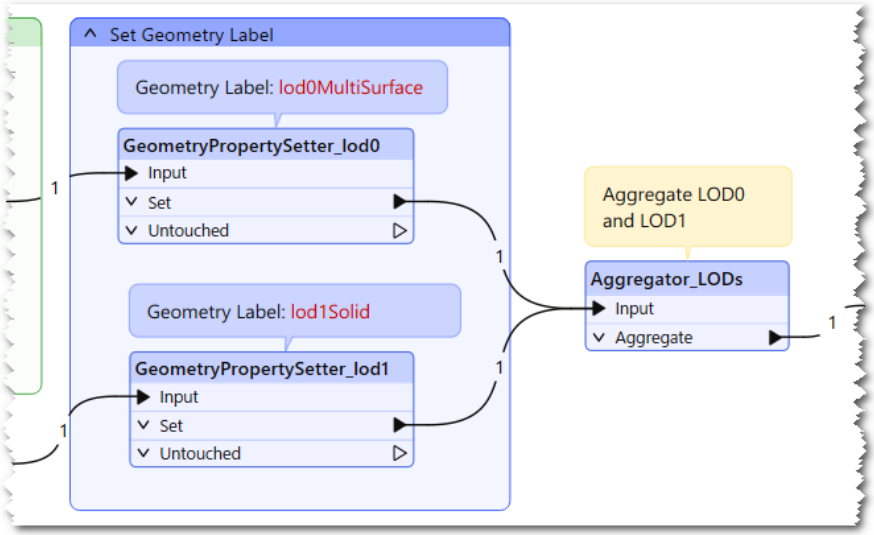

図 1.9 Geometry Label の設定と LOD0, LOD1 の集約

本節で説明した内容のうち、CityGML 3 Writer によって CityGML 3.0 形式のデータを作成するワークスペース作成において知っておくべき共通事項は、次のとおりです。

- マルチスケールモデルを作成する場合は、各 LOD の幾何オブジェクトに対応するジオメトリにそれぞれの関連役割名を Geometry Label として設定した後、Aggregator によって集約する。

1.5 ワークスペース例 1-03: 内部 LOD0

ワークスペース	1-03_interior-lod0.fmw
入力データ	source/1_lod0-1/room_floor.shp（各部屋の床面） source/1_lod0-1/relations.csv（階層構造におけるフィーチャー間の関係定義）
変換結果	output/1-03_interior-lod0.gml

このワークスペース例は、建築物内部 LOD0（部屋の床面）モデルを作成するものです。

本製品では、「1.1 LOD0, LOD1 の定義等」で説明したように建築物内部（部屋）は Storey（階）ごとにグループ化することにしたので、次のようなフィーチャー間の階層構造を持った XML テキストに符号化する必要があります。

```
<bldg:Building gml:id="bldg_01">
    <gml:name>KIT FZK-Haus</gml:name>
    <bldg:buildingSubdivision>
        <bldg:Storey gml:id="stry_01"> <!-- gml_parent_id = "bldg_01" -->
            <gml:name>Ground Floor</gml:name>
            <bldg:buildingRoom>
                <bldg:BuildingRoom gml:id="room_01"> <!-- gml_parent_id = "stry_01" -->
                    <gml:name>03 Ground Floor - Corridor</gml:name>
                    <core:lod0MultiSurface>
                        <gml:MultiSurface>
                        (略)
                        </gml:MultiSurface>
                    </core:lod0MultiSurface>
                </bldg:BuildingRoom>
            </bldg:buildingRoom>
            <bldg:buildingRoom>
                <bldg:BuildingRoom gml:id="room_02"> <!-- gml_parent_id = "stry_01" -->
                    <gml:name>05 Ground Floor - Bedroom</gml:name>
                    <core:lod0MultiSurface>
                        <gml:MultiSurface>
                        (略)
                        </gml:MultiSurface>
                    </core:lod0MultiSurface>
                </bldg:BuildingRoom>
            </bldg:buildingRoom>
            (略)
        </bldg:Storey>
    </bldg:buildingSubdivision>
</bldg:Building>
```

注: 説明のために gml:id の値は簡略化しました。

CityGML 3 Writer で CityGML 文書を出力する際にこのような階層構造を構築するには、各フィーチャーに、

"gml_parent_id"という名前の属性を付与し、その直上位のフィーチャーの gml:id の値を設定する必要があります。

上記例では、gml:id = "room_01"および"room_02"の BuildingRoom（部屋）の"gml_parent_id"属性にその部屋が属する Storey（階）の gml:id の値"stry_01"を設定し、各 Storey の"gml_parent_id"属性に Building（建築物）の gml:id の値"bldg_01"を設定しておくことにより、変換結果の CityGML 文書ではそれらのフィーチャー間の階層構造が構築されます。

このような階層構造の構築に必要な情報は、ワークスペースの入力データセットから抽出できなければなりません。
階層構造を記述するデータのフォーマットやデータ構造は任意ですが、どんな形式で記述するとしても、階層構造内の各フィーチャーについて、それぞれその直上位のフィーチャーを特定するための情報が記述されているということが、必要かつ十分な条件です。

この例では、出力先 CityGML 文書に記述すべきすべてのフィーチャーに ID（feature_id）を与えるとともに、階層構造における直上位のフィーチャーの ID（parent_id）を設定した表を CSV 形式で作成し、それを記述した"relations.csv"ファイルをワークスペースの入力データとして使用することとしました。

表 1.3 フィーチャー間関係定義表

name	feature_id	parent_id
02 Ground Floor - Kitchen	bldg_BuildingRoom_001	bldg_Storey_001
03 Ground Floor - Corridor	bldg_BuildingRoom_002	bldg_Storey_001
04 Ground Floor - Living Room	bldg_BuildingRoom_003	bldg_Storey_001
05 Ground Floor - Bedroom	bldg_BuildingRoom_004	bldg_Storey_001
06 Ground Floor - Bathroom	bldg_BuildingRoom_005	bldg_Storey_001
07 Ground Floor - Office	bldg_BuildingRoom_006	bldg_Storey_001
Ground Floor	bldg_Storey_001	bldg_Building_001
01 Second Floor - Gallery	bldg_BuildingRoom_007	bldg_Storey_002
Second Floor	bldg_Storey_002	bldg_Building_001
KIT FZK-Haus	bldg_Building_001	

表 1.4 フィーチャー間関係定義表の各列の内容

列名	説明	備考
name	フィーチャーの名称	変換結果 CityGML 文書における gml:name 要素の値として使用する。
feature_id	入力データセット内でのフィーチャー識別子	フィーチャータイプの識別も兼ねて、次の書式とした。 {名前空間略称}_{フィーチャータイプ名}_{連番}
parent_id	フィーチャー間の階層構造における直上位フィーチャーの feature_id の値	Building フィーチャーは階層構造の最上位にあるため、parent_id はない。

入力データセットのうち、Shapefile 形式のデータ（各部屋床面のポリゴン）の属性構成は、表 1.2 のとおりです。

（再掲）表 1.2 建築物外形フットプリント / 各部屋の床面 Shapefile ポリゴンデータの属性

属性名	説明
name	建築物 / 部屋の名称
feature_id	入力データセット内でのフィーチャー識別子: {名前空間略称}_{フィーチャータイプ名}_{連番}
zmin	建築物底面 / 部屋床面の標高
zmax	建築物屋根面 / 部屋天井面の標高（最高点）

「表 1.3 フィーチャー間関係定義表」における部屋（BuildingRoom）の"feature_id"の値は、Shapefile 形式部屋床面データの"feature_id"属性の値と一致させています。

以下、ワークスペース例のデータフローについて説明します。

(1) "gml_id"属性と"gml_parent_id"属性の設定

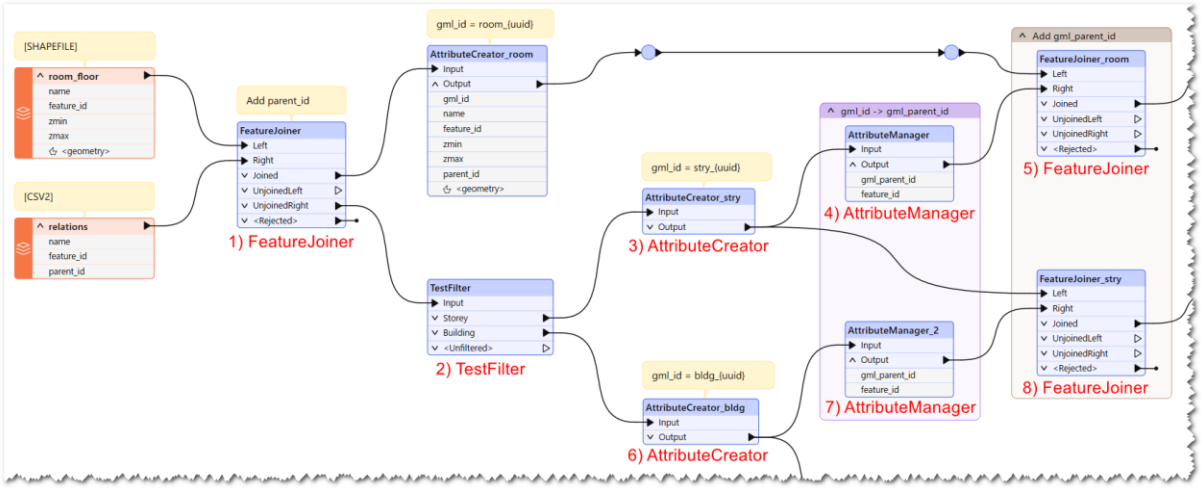

図 1.10 "relations.csv"に基づく階層構造の構築

1) FeatureJoiner: Shapefile リーダーで読み込んだ部屋床面フィーチャーを Left ポートから、CSV リーダーで読み込んだフィーチャー間関係定義表のレコードを Right ポートから入力し、"feature_id"をキーとして結合します。実行すると、Joined ポートからは"parent_id"属性が追加された部屋床面フィーチャーが、UnjoinedRight ポートからは関係定義表レコードのうち Storey、Building に該当するレコードが出力されます。部屋床面フィーチャーにはこの後、"gml_id"属性を付与します（AttributeCreator_room）。
2) TestFilter: FeatureJoiner の UnjoinedRight ポートから出力された関係定義表レコードを、"feature_id"に含まれるフィーチャータイプ名に基づいて Storey と Building に分岐します。以下、分岐後のレコードを Storey フィーチャー、Building フィーチャーと呼びます。
3) AttributeCreator: Storey フィーチャーに"gml_id"属性を追加し、値を設定します。
4) AttributeManager: Storey フィーチャーの"gml_id"属性の名前を"gml_parent_id"に変更します。

5) FeatureJoiner: 部屋床面フィーチャーを Left ポートから、4) で属性名を変更した Storey フィーチャーを Right ポートから入力し、部屋床面フィーチャーの"parent_id"属性、Storey フィーチャーの"feature_id"をキーとして結合することにより、部屋床面フィーチャーに"gml_parent_id"属性（値は 4）で属性名を変更する前の Storey の"gml_id"属性と同じ）を追加します。
6) AttributeCreator: Building フィーチャーに"gml_id"属性を追加し、値を設定します。
7) AttributeManage: Building フィーチャーの"gml_id"属性の名前を"gml_parent_id"に変更します。
8) FeatureJoiner: Storey フィーチャーを Left ポートから、7) で属性名を変更した Building フィーチャーを Right ポートから入力し、Storey フィーチャーの"parent_id"属性、Building フィーチャーの"feature_id"をキーとして結合することにより、Storey フィーチャーに"gml_parent_id"属性（値は 7）で属性名を変更する前の Building の"gml_id"属性と同じ）を追加します。

ここまでで、変換結果 CityGML 文書に書き込む各フィーチャーに、"gml_id"属性、"gml_parent_id"属性（値は階層構造における直上位要素の"gml_id"属性と同じ）が設定できました。

(2) BuildingRoom フィーチャーの出力

部屋床面フィーチャーについては、前節の外部 LOD0 フィーチャーと同じデータフローによって MultiSurface に変換し、CityGML 文書における幾何オブジェクトの関連役割名"lod0MultiSurface"を Geometry Label としてジオメトリのルートノードに設定し、フィーチャーの関連役割名"buildingRoom"を値とする"gml_parent_property"属性を設定してから、CityGML 3 Writer の bldg_BuildingRoom フィーチャータイプから出力します。

ワークスペースへの CityGML 3 Writer およびライターフィーチャータイプの追加手順については、1.3 節で詳説しました。追加するライターフィーチャータイプ名を"bldg_BuildingRoom"に読み替えて適用してください。

(3) Storey, Building フィーチャーの出力

8) FeatureJoiner | Joined ポートから出力された Storey フィーチャー、6) AttributeCreator | Output ポートから出力された Building フィーチャー（どちらもジオメトリはなし）に、それぞれ"gml_parent_property"属性を設定してから、CityGML 3 Writer の bldg_Storey、bldg_Building フィーチャータイプから出力します。
フィーチャータイプ別の"gml_parent_property"属性の値（フィーチャーの関連役割名）は、次のとおりです。

- Storey: buildingSubdivision（Storey の関連役割名）
- Building: cityObjectMember（Building の関連役割名）

CityGML は、ひとつのデータセット（出力先ファイル）に複数のフィーチャータイプを記述できるフォーマットです。この例では、BuildingRoom, Storey, Building フィーチャーを、ひとつのデータセットに出力しようとしています。
BuildingRoom フィーチャー出力用のデータフローを構成したときに CityGML 3 Writer を追加済みなので、Storey フィーチャーと Building フィーチャーを出力するために新たなライターを追加する必要はなく、既存の CityGML 3 Writer に bldg_Storey フィーチャータイプと bldg_Building フィーチャータイプを追加することができます。

既存のライターにフィーチャータイプを追加する方法はいくつかありますが、すでにキャンバス上に当該ライターに属するフィーチャータイプ（この例では bldg_BuildingRoom）が存在する場合は、それを複製（右クリック > Duplicate）し、フィーチャータイプ名を変更するのが最も素早くできます。

キャンバス上の空白部を右クリック > Insert Writer Feature Type で新たなライターフィーチャータイプを追加することもできますが、その場合は、トランスフォーマーと接続する前に、User Attributes のモードを"Automatic"から"Manual"に変更してください

本節で説明した内容のうち、CityGML 3 Writer によって CityGML 3.0 形式のデータを作成するワークスペース作成において知っておくべき共通事項は、次のとおりです。

- 変換結果 CityGML 文書でフィーチャー間の階層構造を構築する場合、階層中の最上位（建築物の場合は Building）を除くすべてのフィーチャーに"gml_parent_id"属性を追加し、それに階層中の直上位のフィーチャーの"gml_id"の値を設定する。
- 複数の異なるタイプのフィーチャーをひとつの CityGML データセットに記述する場合は、CityGML 3 Writer に各タイプに対応するフィーチャータイプを追加して出力する。

CityGML 3 Writer に追加できるフィーチャータイプの種類と数について、CityGML 3.0 のスキーマ（および ADE を使う場合はそのスキーマ）で定義されているフィーチャータイプでなければならないということ以外に制約はないものの、ひとつのフィーチャータイプ数が多くなるとワークスペースが煩雑となり、保守性が低下する可能性があります。

そのような懸念がある場合は、Feature Type Fanout の仕組みを使い、全てまたは一部のタイプのフィーチャーをまとめて単一のライターフィーチャータイプから出力するような構成も可能です。

Feature Type Fanout によるライターフィーチャータイプの構成は、2 章で導入します。

1.6 ワークスペース例 1-04: 外部 LOD0, LOD1, 内部 LOD0

ワークスペース	1-04_exterior-lod0-1_interior-lod0.fmw
入力データ	source/1_lod0-1/building_footprint.shp（建築物外形のフットプリント） source/1_lod0-1/room_floor.shp（各部屋の床面） source/1_lod0-1/relations.csv（階層構造におけるフィーチャー間の関係定義）
変換結果	output/1-04_exterior-lod0-1_interior-lod0.gml

このワークスペース例は、「1.4 ワークスペース例 1-02: 外部 LOD0, LOD1」および「1.5 ワークスペース例 1-03: 内部 LOD0」を統合、一部修正したもので、外部 LOD0 と L0D1 のマルチスケールの建築物モデル、および、Storey ごとにグループ化した LOD0 部屋床面フィーチャーをひとつの CityGML データセットに出力します。

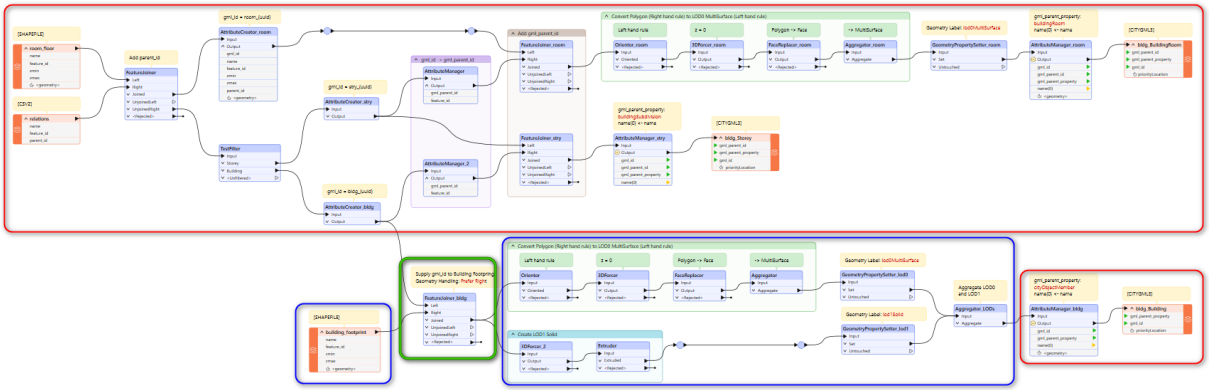

図 1.11 ワークスペース例 1-04_exterior-lod0-1_interior-lod0.fmw の全容

ワークスペースの統合は、元のワークスペースを FME Workbench で開き、キャンバス上で必要な部分（トランスフォーマー等）を選択してコピー&ペーストすることによって行えます。図 1.11 において、青枠内は「1.4 ワークスペース例 1-02: 外部 LOD0, LOD1」から、赤枠内は「1.5 ワークスペース例 1-03: 内部 LOD0」からコピーしたものです。

唯一、緑枠内の FeatureJoiner だけ追加しました。これは、reletions.csv テーブル由来の建築物フィーチャー（ジオメトリなし、"gml_id"属性設定済み）と、building_footprint.shp から読み込んだ建築物外形フットプリントフィーチャーを結合し、後続の LOD0, LOD1 ジオメトリ作成フローに出力します。

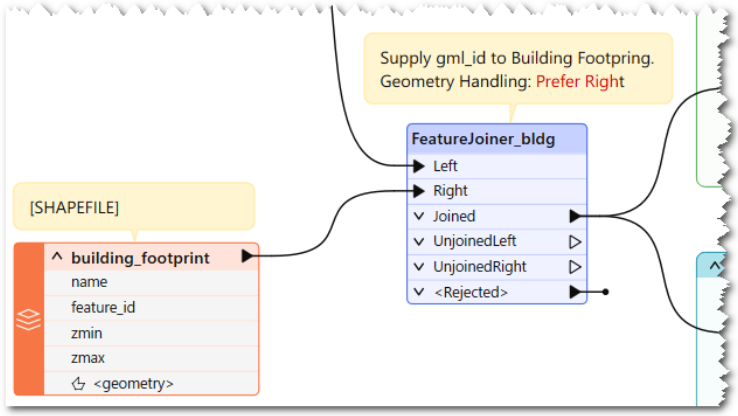

図 1.12 FeatureJoiner: building_footprint.shp から読み込んだフットプリントフィーチャーに gml:id を結合

処理内容の詳細についてはワークスペース例を FME Workbench で開いて確認してください。

本節では、既出のふたつのワークスペース例を統合することにより、変換結果 CityGML 文書におけるジオメトリの種類を統合できることを示しました。統合前の処理を引き継いだだけであって特別に新しいことはなく、CityGML 3.0 形式のデータを作成するワークスペース作成において知っておくべき共通事項の追加はありません。。

CityGML 3 Writer によって CityGML 3.0 形式のデータを作成するワークスペース作成において知っておくべき共通事項として、本章 1.3～1.5 節で挙げた事項をまとめて再掲します。

[1.3 節]

- フィーチャー要素の gml:id 属性の値は、gml_id 属性によって設定する。
- 幾何オブジェクトの関連役割名は、ジオメトリのルートノードに Geometry Label として設定する。
- フィーチャーの関連役割名は、gml_parent_property 属性によって設定する。
- 主題属性要素の値は、対応する属性によって設定する。ただし、XML スキーマ上 2 以上記述し得る主題属性要素の値は、リスト属性によって設定する。
- ライターフィーチャータイプ名、属性名に名前空間プレフィクスをつけるかどうかは、CityGML 3 Writer の Add XML Namespace Prefix to パラメーターによって選択する（本製品が提供するワークスペース例では、フィーチャータイプ名に名前空間プレフィクスをつけ、属性名にはつけない設定で統一した）。

[1.4 節]

- マルチスケールモデルを作成する場合は、各 LOD の幾何オブジェクトに対応するジオメトリにそれぞれの関連役割名を Geometry Label として設定した後、Aggregator によって集約する。

[1.5 節]

- 変換結果 CityGML 文書でフィーチャー間の階層構造を構築する場合、階層中の最上位（建築物の場合は Building）を除くすべてのフィーチャーに"gml_parent_id"属性を追加し、それに階層中の直上位のフィーチャーの"gml_id"の値を設定する。
- 複数の異なるタイプのフィーチャーをひとつの CityGML データセットに記述する場合は、CityGML 3 Writer に各タイプに対応するフィーチャータイプを追加して出力する。

========== Version 1 ここまで ==========