

Building Customization Guide

XMLDB.....	1
1. Building XML files	1
2. Category XML files	4
3. Metrics XML files.....	4
4. Profile effects	4
Clothing for your Sims.....	6
General Naming Guidelines.....	7
<i>Buildings:</i>	7
<i>Figures:</i>	7
<i>Terrain:</i>	7
<i>Effects:</i>	8
<i>Vehicles:</i>	8
<i>Ambients:</i>	8
<i>Pathing Meshes:</i>	8
<i>Portraits:</i>	8
<i>Rubble:</i>	8
<i>Mothball:</i>	9
<i>Security Fence:</i>	9
<i>Texture Archetype:</i>	9
<i>Bridges:</i>	9
<i>Props:</i>	9
<i>Variants:</i>	9
Changing a texture	10
SCSPacks	11
SCSPack File Basics.....	11
Creating an SCSPack file.....	11

XMLDB

1. Building XML files

The main file that defines a building and points to the data needed for that building is placed in the folder [INSTALL ROOT]\Data\XMLDB\Game\Objects\BUILDINGS. The file name can be whatever is desired, though we have been using the convention of "SC5B Building ID.xml" for the sake of consistency. This file will be referred to below as the "building xml".

Within a building xml, a number of xml tags can be added for the purpose of specifying different data related to that building. Here is a list of those tags and their use:

Tag definitions:

<Building>: Primary tag for these files. Defines a new type of Building. The attribute "ID" must be specified and is used by the game to identify the building. ID must be unique for each different building.

Ex. <Building ID="Bank">

...
</Building>

<Type>: Defines the type of object.

Types of buildings are PowerBuilding, FastTravel, Home, Workplace, Venue, and Decoration.

Ex. <Type>Venue</Type>

<Cost>: Numerical value defining the cost in Simoleons to place the building.

Ex. <Cost>100</Cost>

<portrait>: Picture displaying the building in the Building Menu.

Picture files found in: \Data\Graphics\interface\portraits\

Ex. <portrait>SC5BP_Bank</portrait>

<PortraitCard>: Picture displaying the building on its Building Card.

Picture files found in: \Data\Graphics\interface\portraits\

Ex. <PortraitCard>SC5BPC_Bank</PortraitCard>

<Generated_Sim>: This is currently used only for children and works only on buildings with

<SingleFamily>true</SingleFamily> defined. Other Sim Special Sims are spawned from building abilities.

Figure ID can be found in Figures XMLs: \Data\XMLDb\Game\Objects\Figures\

type: Defines the type of Sim that will be spawned.

Flag name can be found in Figures XMLs: \Data\XMLDb\Game\Objects\Figures\

probability: Defines the probability that a Sim will spawn on given day.

Ex. <Generated_Sim type="child" probability="0.5">SC5 Children</Generated_Sim>

<Special_days>: Defines the days of the week on which the building will be open.

Day period choices defined in: [INSTALL ROOT]\Data\XMLDb\Time_Periods.xml

Ex. <Special_days>Weekdays</Special_days>

<Special_hours>: Defines the building's hours of operation.

Open hour periods defined in: [INSTALL ROOT]\Data\XMLDb\Time_Periods.xml

Ex. <Special_hours>Daytime</Special_hours>

<Visit_Duration>: Defines the number of hours Sims will visit a venue.

Visit duration periods defined in: [INSTALL ROOT]\Data\XMLDb\Time_Periods.xml

Ex. <Visit_Duration>Full_Day</Visit_Duration>

<Sim_cash_pay>: Numerical value, defines the amount of money Sims earn for the player each day at work. (This is modified by happiness levels of the Sim.)
 Ex. <Sim_cash_pay>10</Sim_cash_pay>

<Happiness_effects>: Numerical value, defines the amount of happiness Sims will receive on a visit.
 Ex. <Happiness_effects>2</Happiness_effects>

<SingleFamily>: Boolean, defines whether a home houses a single family or multiple families.
 Ex. <SingleFamily>True</SingleFamily>

<Max_population>: Numerical value, defines the number of Sims who can occupy a building simultaneously, or the number of workers that live in a home.
 Ex. <Max_population>10</Max_population>

<MaxEnhancements>: Numerical value, defines the number of enhancements a home can have at one time.
 Ex. <MaxEnhancements>1</MaxEnhancements>

<Operational_Priority>: Numerical value, determines which buildings will be shut off first in an energy deficit.
 Lower number = higher priority
 Ex. <Operational_Priority>1</Operational_Priority>

<Ability>: Defines abilities inherent to a building.
 type: Defines the type of ability.
 Ability types defined in: [INSTALL ROOT]\Data\XMLDb\Abilities.xml. For display purposes, only abilities defined there with the tag <CompositeAbility></CompositeAbility> should be used.
 param: Numerical value, defines one or more parameters regarding the ability.
 Parameter formats documented in: [INSTALL ROOT]\Data\XMLDb\Abilities.xml. This attribute is not needed if a composite ability is used.
 Ex. <Ability type="Big Business" />

<UnlockPrerequisite>: Defines conditions under which a building can be unlocked. Can have multiple of these tags. Specify a type, resource and min or max (amounts).
 "type" can be Energy, Population, Simoleans, Happiness
 Resource defined in: [INSTALL ROOT]\Data\XMLDb\Resources\Resources.xml for "energy"
 Resource for "happiness" is a happiness state defined in [INSTALL ROOT]\Data\XMLDb\Happiness\HappinessCategories.xml. For type "happiness" you must also specify a "subdata" attribute with either "BetterThan" or "WorseThan".
 Ex. <UnlockPrerequisite type="energy" resource="Wealth" min="8" />
 <UnlockPrerequisite type="happiness" resource="Happy" min="8" subdata="BetterThan" />

<HiddenAtStart>: Boolean, this building will not display its attributes until unlock pre-requisites (identical format from above) are achieved. Also adds a number to the Hidden Building counter on the player screen.

<GoalReward>: Boolean, when flagged as true, they need to be linked to completing the archetype pre-requisites in the archetypes.xml
 IE: <GoalReward>true</GoalReward>

<Resource>: Defines the amount of a certain resource given or taken by a building. Two of these tags may exist per building. A third tag may exist, as long as uses the resource "power".
 Resource defined in: [INSTALL ROOT]\Data\XMLDb\Resources.xml
 Quantity can be a positive or negative value.
 Ex. <Resource resource="Wealth" quantity="48" />

<TileWidth>: Numerical value, defines the X coordinate of the Building. Does not alter the appearance of the model, only sets the amount of space the building will take up.

Ex. <TileWidth>9</TileWidth>

<TileHeight>: Numerical value, defines the Y coordinate of the Building. Does not alter the appearance of the model, only sets the amount of space the building will take up.

Ex. <TileHeight>9</TileHeight>

<MiniMapColor>: Defines the building's color on the minimap.

MapColor values can be found in: \Data\XMLDb\UI\MiniMapColors\ColorTable.xml

Ex. <MiniMapColor>Workplace</MiniMapColor>

<UISelectionSet>: Defines the piece of interface that is brought up when the Building is clicked.

Choices include SC5BuildingCardView, SC5SimCard, SC5SpecialSimCard

Ex. <UISelectionSet>SC5BuildingCardView</UISelectionSet>

<Model>: Defines the model that the Building will use

Model files found in: \Data\Graphics\Models\Houses
 \Data\Graphics\Models\Workplaces
 \Data\Graphics\Models\Venues
 \Data\Graphics\Models\Decoration

Ex. <Model>SC5B_Bank</Model>

<SelectedModel>: Defines the model that will act as a selection box for this object.

Model files found in: [INSTALL ROOT]\Data\Graphics\Models\

Ex. <SelectedModel>SC5_Selection_Building_20X20</SelectedModel>

<RecycleRefund>: Numerical value, defines the amount of simoleons reimbursed when the building is deleted.

Ex. <RecycleRefund>1</RecycleRefund>

<CanBeDeleted>: Boolean, defines whether or not the building can be deleted by the player.

Ex. <CanBeDeleted>True</CanBeDeleted>

<CarSet>: Defines the type of cars that residents of a building drive.

Car sets defined in: [INSTALL ROOT]\Data\XMLDb\CarSets.xml

Ex. <CarSet>Classic Cars</CarSet>

<Ground_texture>: Numerical value, defines the texture that will appear below the building.

Choices include -1 (no texture), 0 (dirt), or 1 (grass)

Ex. <Ground_texture>1</Ground_texture>

<ObjectSoundSet>:

SoundSet values can be found in: [INSTALL ROOT]\Data\XMLDb\Sound\Sound Sets\SoundSets.xml

Ex. <ObjectSoundSet>default</ObjectSoundSet>

<PropSet>: Defines prop set for this building.

Prop set values can be found in: [INSTALL ROOT]\Data\XMLDb\Art\Props\Props.xml

Ex. <PropSet>Mothball 10x10</PropSet>

<RubbleModelFinal>: Defines the model that will appear when the building has finished burning down.

Rubble models can be found in: [INSTALL ROOT] \Data\Graphics\Textures\

Ex. <RubbleModelFinal>SC5BC_10x10</RubbleModelFinal>

<BurntModel>: Defines the model that will appear at a building that is burning.

Burnt models can be found in: [INSTALL ROOT]\Data\Graphics\Textures\

Ex. <BurntModel>SC5BC_10x10</BurntModel>

<PathThrough>: Boolean value that determines whether or not a Sim from the outside can take a "short cut" through the building to get to another destination.

Ex. <PathThrough="True"/> means Sims will take short cuts through the building

Ex. <PathThrough="False"/> means Sims cannot treat the building node as a valid waypoint for their polypath calculations.

<Carbon>: Defines the amount of carbon produced by the building.

Ex. <Carbon>150</Carbon>

2. Category XML files

In order to associate a building with other buildings, the building must be placed in the appropriate category xml files. Categories are used to group buildings in terms of how they are affected in the game by building abilities and actions, special Sims, and events. A specific subset of building categories are used to determine which cultural graphics are applied to buildings as the city evolves.

Category XML files are located in the [INSTALL ROOT]\Data\XMLDB\Categories\ folder. Existing category files are defined as either gameplay categories or profile categories. Gameplay categories define text string ID's and icons that are used to display the category on the building's card in game.

Ex. <Category ID="Corporate" stringid="CATEGORY_CORPORATE" iconoffset= "2">

3. Metrics XML files

All buildings in the game are associated with one or more societies. These associations are defined in metrics XML files, which are located in the Master\Data\XMLDB\Game\Metrics\ folder.

The metric is defined at the start of the metric file.

Ex. <Metric ID="Capitalist">

Within the metric, any building can be given a value for the metric which measures how strongly the building is associated with that particular society. Each building is referenced by its building ID and given a numeric value, known as 'contribution' which is the measure of how the building relates to the society.

Ex. <Building BuildingID="Bank" Contribution="15"></Building>

These values can be positive, negative, zero, or left blank. Positive values indicate that the building is associated with the society and negative ones indicate that the buildings are in opposition to what the society represents. Zero or no defined value means the building is essentially neutral with respect to that particular society (as does having no entry at all in the metric). In order for the building to display in the list of buildings in the build menu when a particular society filter is selected, it must have a value of at least 1 in the appropriate metric file.

4. Profile effects

The sum of all metrics in the city are referenced against Profile files located in the [INSTALL ROOT]\Data\XMLDB\Game\ Profiles. The values created by the cumulative effects of the metrics are referenced by these files to then call into effect the graphic, audio, and other effects desired for that profile.

Profile categories are used solely for determining cultural graphics and do not affect gameplay or display on the building's card in game. Profile categories are defined as such when the id for the category is defined, and do not define any sting or icon information. Existing profile categories are all named with the prefix "P_" in order to distinguish them from the gameplay categories, but this nomenclature is not required for the category to function properly, only the definition in the ID.

Ex. <Category ID="P_Agriculture" profilecategory="true">

Buildings can be (and often are) part of more than one category. To associate a building with a particular category, a new entry must be added to the category file which references the building's ID (from the specific building's XML file).

Ex. <Building ID="Law Offices"/>

Almost all categories are used exclusively for buildings, but there are two special categories that are used for some pathfinding exceptions for certain groups of special Sims. Pathfinding_IgnoresFires.xml is used to define which Sims do not flee from fires. Pathfinding_NoMassTransit.xml is used to define which Sims cannot make use of Mass Transit buildings, such as bus stations and subways.

Needed for Buildings (models textures appropriate xmls for props):

[INSTALL ROOT] \data\graphics\Models

Building_model.xac

Bulilding_model_FLP.xac

\master\data\graphics\Textures

Clothing for your Sims

A. To add new clothing textures within existing Clothing Sets (i.e. a new outfit within the “Blue Collar” clothing set):

1. Additional texture files go into Textures/clothing directory
2. Additional <Outfit> statements added to [ClothingSet.xml](#)

B. To add new Clothing Sets for Sims (i.e. an “Iron Chef” clothing set):

1. Additional texture files go into Textures/clothing directory
2. Additional <ClothingSet> statements and <Outfit> substatements added to [ClothingSet.xml](#)
3. Changes to relevant [Building XML](#) files to call the new Clothing Set (i.e. the “Kitchen Stadium” building needs to specify that workers should be textured from the “Iron Chef” clothing set)

This is assuming that the new textures will fit well on existing Sim models. In some cases, that might not be true, and we would also need to add new Sim meshes. (For example – Sim “evening gown” might require a model with an ankle-length dress in the geometry.)

C. To add new Sim models (in addition to [A] and/or [B] above) :

1. New Sim model goes into Models/Figures
2. New entry for the model in [ModelSets.xml](#)
3. New entry for the model in [ModelAnimations.xml](#) to specify the animations it uses

D. For new animations for Sims:

1. New animations go into Models/Figures
2. New entries for those animations go into [AnimationSet.xml](#)
3. If those animations have variants, or are variants, new entries go into [Animations.xml](#)
4. If those animations have props, new entries go into [ModelAnimations.xml](#)

New Special Sim

A. New special sim .cs files(s)

B. New special sim .xml file in Xmlldb\Game\Objects\Figures folder

General Naming Guidelines

Buildings:

Models:

An example of the model naming convention is simply:

[Game Name][B for Building]_[Building Name].XAC, for example:

SC5B_Townhouse.XAC

Textures:

For Textures the same naming convention applies, it should mirror the name of the model, with the exception of course that its file format is .dds

Figures:

Models:

An example of the model naming conventions is as follows:

Citizen Sims:

[Prefix]_[Sex of Character]_[AGE=Character Name]_[Variant]_[Model Designation]. XAC, for example:

SC5_M_Adult_1_M.XAC

Special Sims:

[Prefix]_[Sex of Character]_[Character Name]_[Model Designation].XAC, for example:

SC5_M_Astronaut_M.XAC

Animations:

For Animations that coincide with this character, the convention looks like the following:

[Prefix]_[Sex of Character]_[Character Name]_[Animation Name]_[Animation Designation].XSM, for example:

SC5_M_Astronaut_walk_A.XSM

Textures:

Since the textures will be used on multiple models, the texture names for the figure models will be as follows, in order to not lock it down to a specific model:

SC5[Sex][Component*]_[Hairstyle/Description].dds

Examples:

SC5MC_Tshirt_bluejeans.dds
SC5FC_Dress.dds

SC5MS_bald_white.dds
SC5FS_brown_dark.dds

Terrain:

An example of the terrain texture naming convention is as follows:

[Prefix]_[Terrain Name]_[Iteration]_[Shader].dds, for example:

SC5T_Plaza_1_bump.dds

The only time this system is different is with diffuse textures, which simply drops the Shader portion off of the name, such as:

SC5T_Plaza_1.dds

Effects:

An example of the effects texture naming convention is as follows:

[Effect Designation]_[Effect Name].dds, for example:

Sfx_smoke.dds

Vehicles:

An example of the vehicle naming convention is as follows:

[Prefix][VehicleFlag]_[Vehicle Type]_[Model or Animation].XAC, for example:

SC5V_Horse_M.XAC (In the case for models)

SC5V_Horse_walk_A.XSM (In the case for animations)

Ambients:

An example of the ambient naming convention is as follows:

[Prefix][AmbientFlag]_[Ambient Name].XAC

SC5A_Candycanelamppost.XAC

Texture will mirror model name.

Pathing Meshes:

An example of the Pathing Mesh naming convention is as follows:

[Prefix][Building Flag]_[Building Name]_[Pathing Mesh Flag].XAC

SC5B_Farm_House_FLP.XAC

Portraits:

An example of the Building/Figure naming convention is as follows:

[Prefix][Building/Figure Flag][Portrait Flag]_[Building Name].tga

SC5BP_MovieTheatre.tga

For the larger portraits that will be featured in the Building Card, the naming convention for that is as follows:

[Prefix][Building/Figure Flag][Portrait Flag][Card Flag]_[Building Name].tga

SC5BPC_MovieTheatre.tga

Rubble:

An example of the rubble naming convention is as follows:

[Prefix][Building Flag]_[Collapse]_[Area Covered].XAC, for example:

SC5BC_4x4.XAC

Mothball:

An example of the rubble naming convention is as follows:

SC5[MothBall]_[TYPE]_[Piece][Variant-if any].XAC

SC5MB_Woodplank_Straight2.xac

SC5MB_Woodplank_Corner.xac

Security Fence:

An example of the rubble naming convention is as follows:

SC5[SecurityFence]_[TYPE]_[Piece][Variant-if any].XAC

SC5SF_Woodplank_Straight1.xac

SC5SF_Woodplank_Corner.xac

Texture Archetype:

Archetype- (Funland, Orwellian)

SC5B_[buildingname]_[Archetype].dds

SC5B_Corner_Deli_Funland.dds

Bridges:

SC5BR_[Material]_[Type]

Ex. SC5BR_Stone_Arch

Props:

SC5PC_[Object] (Prop Cultural)

SC5PR_[Object] (Prop Road)

SC5PF_[Object] (Prop Figure)

Ex. SC5PC_Airconditioner.xac

Variants:

SC5B_[Building]_[Variant].dds

SC5B_Townhouse_B or _C

Ex. SC5B_Airconditioner_B.xac/.dds

Variant Portraits

SC5BP_[Building]_[Variant].tga

SC5BP_Townhouse_B or _C

Ex. SC5BP_Homestead_B.tga

Changing a texture

Here is an example of taking the texture of one object and changing it to another, in this case taking a 'blue Bank' and making it green.

Create Extra Texture: SC5B_Bank_Green

Create copy of Bank xac : SC5B_Bank_Green_Roof.xac

Create copy of Bank xml: SC5b Green Bank.xml

Modify building ID to <Building ID="BankGr">

Modify model to <Model> SC5B_Bank_Green_Roof </Model>

[INSTALL ROOT]\data\xml\Game\Cultural Graphics Effects

Green Roof Bank Set.xml

Create Green Cultural Graphics Set with this info: (note keep this called "Custom Textures")

```
<CulturalGraphicSets>
```

```
<GraphicSet name="Custom Textures">
```

```
<Building name=" BankGr " texture="SC5B_Bank_Green"> </Building>
```

```
</GraphicSet>
```

```
</CulturalGraphicSets>
```

SCS Packs

SCSPack File Basics

SCSPacks are used to transfer content to another user. An SCSPack file is a single file in a compressed format containing the needed game files to add content. Usually these will consist of XMLs, XACs, XSMs, FLPs, LODs, and other art assets.

When the game installs assets from a SCSPack file they go into the local computer's "\Documents and Settings\All Users\Application Data\SimCity Societies\Data" folder. No content is placed in the SCS install directory. This allows you to wipe out the installed SCSPack directory in the event something goes wrong.

It is important to note that the game engine first looks for content in the "All Users" location before the installed SCS directory. So user created content would take precedence over core game content.

Creating an SCSPack file

At heart an SCSPack file is an XML file detailing the directory structure in which to place all of the relevant game files. An example can be found below. All directories in this XML are relative to "\Documents and Settings\All Users\Application Data\SimCity Societies\"

```
<?xml version="1.0" encoding="utf-8"?>
<Package>
<FileAdd name="Data\Graphics\Models\Houses\YOUR_BUILDING.xac" />
<FileAdd name="Data\Graphics\Textures\interface\portraits\MY_PACKBP_YOUR_BUILDING" />
<FileAdd name="Data\Graphics\Textures\interface\portraits\MY_PACKBPC_YOUR_BUILDING.png" />
<FileAdd name="Data\XMLDb\Categories\MY_PACK All.xml" />
<FileAdd name="Data\XMLDb\Game\Objects\BUILDINGS\MY_PACK YOUR_BUILDING.xml" />
<FileAdd name="Localization\en_us\ObjectText\MY_PACK BuildingText.xml" />
</Package>
```

1. Create the SCSPack XML file which should list each file (similar to the example above)
2. Place your SCSPack file in "\Documents and Settings\YOUR_USER_NAME\My Documents\SimCity Societies\Export"
3. Make sure all of the files you referenced in your SCSPack XML are in one of two places. The installed game Data directory or "\Documents and Settings\All Users\Application Data\SimCity Societies\Data" (the latter should mimic the directory structure in the installed directory and your XML file from Step 1).
4. Run the game and it will attempt to package together all of the files you listed in your XML (assuming they are in one of the two locations listed in the previous line) and the XML itself into a file with the same name but a different extension.
5. Look in your Export directory the game will have changed "YOUR_PACK_FILE_NAME.xml" to "YOUR_PACK_FILE_NAME.SCSPack" if all went well.

Congratulations you just made a SCSPack!

INSTALL- To "Install" your pack, double click on the pack and it will disappear. This actually places it in the proper location to be unpacked by the executable when the game is run.

© 2007 Electronic Arts Inc. EA, the EA logo and SimCity are trademarks or registered trademarks of Electronic Arts Inc. in the U.S. and/or other countries. All Rights Reserved. Tilted Mill Game Engine © 2007 Tilted Mill Entertainment, Inc. All rights reserved. All other trademarks are the property of their respective owners.