QOLCities Specification

Version 1.0

Index

A cities building blocks

Cities can be composed of custom resources, needs, indicators, projects, population and listeners.

Resources are used to construct or operate buildings/actions. If a building requires one wood during it’s build phase no progress can be made unless the city has access to 1 wood. Resources do not accumulate over multiple turns and are not consumed after being used.

Needs show what the city must have to run smoothly. They have a requirement target and a provided value. Providing less than what is required leaves the need unfulfilled.

Indicators are a view on the society and it’s culture. As an example, a believe system could be founded on science or religion. Each citizen will have an affinity towards science and towards religion. They are not mutually exclusive but the stronger affinity will define what believe they accept. Those believes could be Darwin or Creationism. In addition each person could be subject to suppression which would lower their affinity – if any. Which means in a society that is deeply religious, a scientific mind would feel oppressed and might bend to Creationism due to that pressure.

Projects are the most flexible parts in a city. They are used to represent buildings, districts, achievements, crisis and can support many more ideas. Projects can exists multiple times in a city using the same configuration entry. They also provide actions to the city, which exist at most once per configuration entry.

The population governs who is allowed to be a citizen and what each citizen contributes to the city. Maybe your city is only allowing cats and requires balls of yarn to be built for each cat to avoid a cat-tastrophy.

To expand the modding capabilities listeners can be configured on most objects to implement custom functionality that can be triggered at varying times in the workflow.

	Project	Action	City	Resource	Need	Indicator	Listener
Configuration							
Savestate							
Instances	0-N	0-1	1	1	1	1	0-N
Listenersupport							

Table 1: Overview of objects and their capabilities

Configuration is everything written in xml. Objects with a savestate can preserve information in the savegame. Instances means how many objects can exist for each configuration.

Standard events

In order to tie new code into the processing of the city an event engine is being used. These are events raised by the city's base class. Custom events can be used as well either for extending the frameworks functionality or for better organizing/ordering custom code in bulk.

Simulation		**Refresh cycle**	
Tick		PreCalculateStaticContributions	
	DailyTick		*Refresh cycle*
	Refresh cycle	PostCalculateStaticContributions	
	ConsumeContributions	PreCalculateDynamicContributions	
		PostCalculateDynamicContributions	
Project added		**Project building**	
PreProjectAdded		PreProjectAddingLabour	
PostProjectAdded		PostProjectAddingLabour	
		PostProjectBuild	

Table 2: Standard event chains

Tick : QOLCityEvent

This event is raised for each game turn.

DailyTick : QOLCityEvent

This event is raised for each 60000 game turns(configurable). Pacing for the cities simulation is tied to this event while pacing for external task like incident delays is tied to the game turns.

PreCalculateStaticContributions : QOLCityEvent_State

Whenever the city refreshes its state this event is raised. It carries the current and soon to be previous city contributions as payload. This event can be used when calculating tendencies in the forecast. This means code mainly **reads** the contributions.

This is also the only event in the refresh cycle that may trigger another refresh.

PostCalculateStaticContributions : QOLCityEvent_State

After the new contributions have been filled with the default configured contributors(projects, population, indicators) and assigned to the city in a refresh cycle, this event is raised with the contributions as payload. This event can be used to add fixed contributions before other code evaluates the available contributions. This means code mainly **writes** contributions.

PreCalculateDynamicContributions : QOLCityEvent_State

PostCalculateDynamicContributions : QOLCityEvent_State

Both events serve the same purpose with the pre event being raised just before the post event as the final step in a refresh cycle. Dynamic contributions read what is available and generate contributions based on that. Converting labour into resources would be an example. If the labour from static contributions is not sufficient the resource gain might be lowered.

ConsumeContributions : QOLCityEvent

This event is only raised on a daily tick after the automatic spawning and all refresh events took place. The difference between the refresh cycle and this one is that changes made here do not show up in the forecast and changes made here occur only once every daily tick. This is used to build projects or repair items.

PreProjectAdded : QOLCityEvent_Project

This event is raised right before a project is added to the city. The project's listeners have not been registered yet so a project can not *see* itself with this event.

PostProjectAdded : QOLCityEvent_Project

This event is raised after a project was added to the city and it's listeners were registered.

PreProjectAddingLabour : QOLCityEvent_Project

This event is raised before labour is added to a project when building it.

PostProjectAddingLabour : QOLCityEvent_Project

This event is raised after labour was added to a project when building it.

PostProjectBuild : QOLCityEvent_Project

This event is raised after enough labour was added to a project to complete the building of it. This is exclusive to projects built with labour, if a project spawns built this event won't occur.

Your first city

The goal will be to model a representation of vanilla gameplay in a city. Create an empty xml mod to load the def files. Then add a new xml file in your defs folder and add an entry for the city configuration. The final result of this exercise is also available as a zip file alongside this document.

```
<?xml version="1.0" encoding="utf-8"?>
<Defs>
	<RIMMSqol.remnantcolony.QOLCityDef>
		<defName>MyCity</defName>
		<label>my city</label>
```

```
            <description>My first city.</description>
      </RIMMSqol.remnantcolony.QOLCityDef>
</Defs>
```

First we have to link the city to a world object. That object will represent the city on the world map. It also defines the class that will be used to save the city. Defining a custom class allows changing the citie's behaviour and more importantly the GUI. We use the default city **WorldObjectQOLCity**. We can also define what the city is willing to exchange by declaring a trader kind. This has to cover all products that the city places in storage and all products that are expected to be placed in storage by the settlement. Unless something shouldn't be traded we can simply allow all. Stockgenerators of the trader will not be used to spawn a default inventory.

```
<RIMMSqol.remnantcolony.QOLCityDef>
      …
      <worldObject>WorldObjectQOLCity</worldObject>
      <traderKind>RemnantColony</traderKind>
</RIMMSqol.remnantcolony.QOLCityDef>
```

Now it is time to allow people into the city. For that we need to configure the population. We make sure that no animals can become citizen by requiring at least human-like intelligence.

```
<RIMMSqol.remnantcolony.QOLCityDef>
      …
      <population>
            <socialStandings>
                  <li>QOLSSNormal</li>
            </socialStandings>
            <immigrationLaws>
                  <li
Class="RIMMSqol.remnantcolony.QOLCityPawnFilter_AtLeastHumanlikeIntelligence"></
li>
            </immigrationLaws>
      </population>
</RIMMSqol.remnantcolony.QOLCityDef>
```

Now the citizens will need to be cared for like they would in a settlement. Food and shelter are the basic needs. We model them by declaring a need for shelter, a resource for raw food and a need for meals. Then we need a bedroom and kitchen building to satisfy those needs.

```
<RIMMSqol.remnantcolony.QOLCityDef>
      …
      <projects>
            <li>MyCityBedroom</li>
            <li>MyCityKitchen</li>
      </projects>

      <needs>
            <li>MyCityHousing</li>
            <li>MyCityMeals</li>
      </needs>

      <resources>
            <li>MyCityFood</li>
      </resources>
</RIMMSqol.remnantcolony.QOLCityDef>
```

Now it is necessary to have people create those needs. All biological citizens will require one meal per day. All citizens will require one unit of housing per day.

```
<RIMMSqol.remnantcolony.QOLCityDef>
	…
	<population>
		…
		<contributions>
			<li>
				<pawnFilters>
					<li
Class="RIMMSqol.remnantcolony.QOLCityPawnFilter_Flesh"/>
				</pawnFilters>
				<multiplierPopulationCount>
					1.0
				</multiplierPopulationCount>
				<needs>
					<li>
						<def>MyCityMeals</def>
						<provided/>
						<required>
							<flat>1.0</flat>
							<factor>0.0</factor>
						</required>
					</li>
				</needs>
			</li>
			<li>
				<multiplierPopulationCount>
					1.0
				</multiplierPopulationCount>
				<needs>
					<li>
						<def>MyCityHousing</def>
						<provided/>
						<required>
							<flat>1.0</flat>
							<factor>0.0</factor>
						</required>
					</li>
				</needs>
			</li>
		</contributions>
	</population>
</RIMMSqol.remnantcolony.QOLCityDef>
```

In order to build the required buildings we now need to define a labour need whose surplus is used to perform non upkeep jobs in the city. All citizens will provide labour. We also define the listener that is doing the actual work of building projects consuming surplus labour in the process.

```
<RIMMSqol.remnantcolony.QOLCityDef>
	…
	<needs>
		…
		<li>MyCityLabour</li>
	</needs>

	<labourDef>MyCityLabour</labourDef>

	<population>
```

```
            …
            <contributions>
                  …
                  <li>
                        <multiplierPopulationCount>
                              1.0
                        </multiplierPopulationCount>
                        <needs>
                              <li>
                                    <def>MyCityLabour</def>
                                    <provided>
                                          <flat>5.0</flat>
                                          <factor>0.0</factor>
                                    </provided>
                              </li>
                        </needs>
                  </li>
            </contributions>
      </population>

      <listeners>
            <li>QOLBuildProjects</li>
      </listeners>
</RIMMSqol.remnantcolony.QOLCityDef>
```

Right now we have needs but no downsides to leaving them unsatisfied. So the player could just ignore them. To remedy this we use crisis projects.

```
<RIMMSqol.remnantcolony.QOLCityDef>
      …
      <projects>
            …
            <li>MyCityHomeless</li>
            <li>MyCityStarvation</li>
            <li>MyCityOvertime</li>
      </projects>
</RIMMSqol.remnantcolony.QOLCityDef>
```

With that we can move on to defining the projects, needs and resources we have used so far. Needs are the easiest part.

```
<RIMMSqol.remnantcolony.QOLCityNeedDef>
      <defName>MyCityHousing</defName>
      <label>housing</label>
      <description>Demand for housing.</description>
      <order>10</order>
</RIMMSqol.remnantcolony.QOLCityNeedDef>

<RIMMSqol.remnantcolony.QOLCityNeedDef>
      <defName>MyCityMeals</defName>
      <label>meals</label>
      <description>Demand for meals.</description>
      <order>20</order>
</RIMMSqol.remnantcolony.QOLCityNeedDef>

<RIMMSqol.remnantcolony.QOLCityNeedDef>
      <defName>MyCityLabour</defName>
      <label>labour</label>
      <description>Available labour.</description>
      <order>30</order>
</RIMMSqol.remnantcolony.QOLCityNeedDef>
```

The resource *MyCityFood* is representing raw food that the city can then convert into meals. Resources are either produced by the city or supplied by the player in form of boosters that consume some items from storage. We allow any raw food to be converted into food supplies for the city.

```
<RIMMSqol.remnantcolony.QOLCityResourceDef>
      <defName>MyCityFood</defName>
      <label>raw food</label>
      <description>How much raw food is accessible.</description>
      <callback
Class="RIMMSqol.remnantcolony.callbacks.QOLCityCallback_ResourceBooster">
            <thingCategory>FoodRaw</thingCategory>
            <amount>20</amount>
            <boosterProject>MyCityBoostFood</boosterProject>
      </callback>

      <icon>UI/Icons/ThingCategories/FoodRaw</icon>
      <order>10</order>
</RIMMSqol.remnantcolony.QOLCityResourceDef>
```

The booster is represented as a project. That project can not be built and is instead spawned via the resource callback. Since its not a permanent solution it will despawn after a certain time.

```
<RIMMSqol.remnantcolony.QOLCityProjectDef Abstract="True" Name="MyCityBooster" >
      <order>-1</order>
      <lifetime>120000</lifetime>
      <autospawnTimer>0</autospawnTimer>
      <cancellable>false</cancellable>
      <toggleable>false</toggleable>
      <labour>0</labour>
      <maximumInstances>99</maximumInstances>
      <worldObject IsNull="True"/>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef ParentName="MyCityBooster">
      <defName>MyCityBoostFood</defName>
      <label>booster food</label>
      <description>Provides raw food that can be used by the city.</description>
      <states>
            <active>
                  <resources>
                        <li>
                              <def>MyCityFood</def>
                              <flat>1.0</flat>
                        </li>
                  </resources>
            </active>
      </states>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.QOLCityDef>
      …
      <projects>
            …
            <li>MyCityBoostFood</li>
      </projects>
</RIMMSqol.remnantcolony.QOLCityDef>
```

The next project is the bedroom, it is a building that will be built by the city. It won't time out and it is presented to the user in a specific category. That category dictates how the project is visualized and what actions are available for it.

```
<RIMMSqol.remnantcolony.QOLCityProjectDef>
	<defName>MyCityBedroom</defName>
	<label>bedroom</label>
	<description>Provides living space for one colonist.</description>
	<order>10</order>
	<cancellable>true</cancellable>
	<toggleable>false</toggleable>
	<labour>5</labour>
	<category>MyCityBuildings</category>
	<maximumInstances>0</maximumInstances>
	<states>
		<active>
			<needs>
				<li>
					<def>MyCityHousing</def>
					<provided>
						<flat>1.0</flat>
					</provided>
				</li>
			</needs>
		</active>
	</states>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.QOLCityProjectCategoryDef>
	<defName>MyCityBuildings</defName>
	<label>Buildings</label>
	<description>Projects located in the city.</description>
	<order>10</order>
	<renderer>RIMMSqol.remnantcolony.cities.QOLCityProjectsRenderer</renderer>
</RIMMSqol.remnantcolony.QOLCityProjectCategoryDef>
```

With the kitchen we want to add the ability to prepare meals from raw food spending labour in the process. Essentially the kitchen will be a workbench and we assign labour and resources to a recipe that creates meals. In order to achieve that we won't use the static contributions in the project's states but an action. The action will provide a configuration dialog where the user can decide how many meals are cooked at most, with multiple kitchens increasing the upper limit. Then the listener on the action will calculate how many meals can be produced and are needed. With that the input and output of the meal recipe are applied for each production cycle. The listeners order is crucial in this, since it defines which labour consumer has priority. That way the labour goes to food production either before building new projects or after.

```
<RIMMSqol.remnantcolony.QOLCityProjectDef>
	<defName>MyCityKitchen</defName>
	<label>kitchen</label>
	<description>Provides tools and space to prepare meals from raw
food.</description>
	<order>20</order>
	<cancellable>true</cancellable>
	<toggleable>true</toggleable>
	<labour>10</labour>
	<category>MyCityBuildings</category>
	<maximumInstances>10</maximumInstances>
```

```
      <actions>
            <li>MyCityActionCook</li>
      </actions>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.actions.QOLCityActionDef>
      <defName>MyCityActionCook</defName>
      <label>cook meals</label>
      <description>Prepares meals from raw food.</description>
      <icon>Things/Item/Meal/Simple</icon>
      <order>10</order>
      <action
Class="RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_Convert">
            <inputPerCycle>
                  <resources>
                        <li>
                              <def>MyCityFood</def>
                              <flat>1.0</flat>
                        </li>
                  </resources>
            </inputPerCycle>
            <outputPerCycle>
                  <needs>
                        <li>
                              <def>MyCityMeals</def>
                              <provided>
                                    <flat>4.0</flat>
                              </provided>
                        </li>
                  </needs>
            </outputPerCycle>
            <labourPerCycle>1</labourPerCycle>
            <maxCyclePerInstance>10</maxCyclePerInstance>
            <onlyNecessaryProduction>true</onlyNecessaryProduction>
      </action>
      <listeners>
            <li>MyCityConverter</li>
      </listeners>
</RIMMSqol.remnantcolony.actions.QOLCityActionDef>

<RIMMSqol.remnantcolony.listeners.QOLCityListenerDef>
      <defName>MyCityConverter</defName>
      <label>cook meals</label>
      <description>This listener converts input contributions to output
contribution consuming labour in the process.</description>
      <eventIds>
            <li>
                  <eventId>PreCalculateDynamicContributions</eventId>
                  <order>1000</order>
            </li>
      </eventIds>

<listenerClass>RIMMSqol.remnantcolony.listeners.QOLCityListener_Conversion</list
enerClass>
</RIMMSqol.remnantcolony.listeners.QOLCityListenerDef>
```

The three crisis projects will be identical but for the labelling. They all will have a progress bar that show the progress from 0 to 10. Internally the progress is tracked for up to 20 to avoid the ability of stopping a long escalating crisis immediately. Every day a crisis decays by a fixed amount resolving itself if nothing contributes to it. If certain prerequisites are

fulfilled, in these cases a need is negative, then the progress is increased. Once the threshold of 10 has been reached citizens are at risk of dying due to disease.

```
<HediffDef>
	<defName>MyCityDisease</defName>
	<label>disease</label>
</HediffDef>

<RIMMSqol.remnantcolony.QOLCityProjectCategoryDef>
	<defName>MyCityCrisis</defName>
	<label>Crisis</label>
	<description>Situations that need to be monitored to prevent or encourage
events.</description>
	<order>20</order>
	<renderer>RIMMSqol.remnantcolony.cities.QOLCityCrisisRenderer</renderer>
</RIMMSqol.remnantcolony.QOLCityProjectCategoryDef>

<RIMMSqol.remnantcolony.listeners.QOLCityListenerDef>
	<defName>MyCityCrisisProgression</defName>
	<label>progress crisis</label>
	<description>This listener will apply the daily decrease and progression
configured in QOLCityProjectDef_Crisis.</description>
	<eventIds>
		<li>
			<eventId>ConsumeContributions</eventId>
			<order>100</order>
		</li>
	</eventIds>

<listenerClass>RIMMSqol.remnantcolony.listeners.QOLCityListener_CrisisProgressio
n</listenerClass>
</RIMMSqol.remnantcolony.listeners.QOLCityListenerDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef Abstract="True" Name="MyCityCrisis">
	<order>10000</order>
	<lifetime>0</lifetime>
	<autospawnTimer>1</autospawnTimer>
	<cancellable>false</cancellable>
	<labour>0</labour>
	<category>MyCityCrisis</category>
	<maximumInstances>1</maximumInstances>
	<sendCompletedNotification>false</sendCompletedNotification>

<stateClass>RIMMSqol.remnantcolony.projects.QOLCityProject_Crisis</stateClass>
	<dailyDecrease>1</dailyDecrease>
	<minProgression>0</minProgression>
	<maxProgression>20</maxProgression>
	<minDisplayProgression>0</minDisplayProgression>
	<maxDisplayProgression>10</maxDisplayProgression>
	<listeners>
		<li>MyCityCrisisProgression</li>
	</listeners>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef ParentName="MyCityCrisis"
Class="RIMMSqol.remnantcolony.projects.QOLCityProjectDef_Crisis">
	<defName>MyCityHomeless</defName>
	<label>homelessness</label>
	<description>A lack of shelter exposes people to health
risks.</description>
	<progression>
		<li>
			<requirements>
```

```
                        <li
Class="RIMMSqol.remnantcolony.requirements.QOLCityRequirement_NeedValueIsNegativ
e">
                              <need>MyCityHousing</need>
                        </li>
                  </requirements>
                  <dailyProgress>2</dailyProgress>
            </li>
      </progression>
      <triggers>
            <li>
                  <requirements>
                        <li
Class="RIMMSqol.remnantcolony.requirements.QOLCityRequirement_CrisisProgressIsAb
ove">
                              <crisis>MyCityHomeless</crisis>
                              <value>10.0</value>
                        </li>
                  </requirements>
                  <trigger
Class="RIMMSqol.remnantcolony.triggers.QOLCityTrigger_KillPopulation">
                        <numOfPeopleMax>1</numOfPeopleMax>
                        <numOfPeopleMin>1</numOfPeopleMin>
                        <hediff>MyCityDisease</hediff>
                        <chance>0.05</chance>
                  </trigger>
            </li>
      </triggers>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef ParentName="MyCityCrisis"
Class="RIMMSqol.remnantcolony.projects.QOLCityProjectDef_Crisis">
      <defName>MyCityStarvation</defName>
      <label>starvation</label>
      <description>A lack of food exposes people to health risks.</description>
      <progression>
            <li>
                  <requirements>
                        <li
Class="RIMMSqol.remnantcolony.requirements.QOLCityRequirement_NeedValueIsNegativ
e">
                              <need>MyCityMeals</need>
                        </li>
                  </requirements>
                  <dailyProgress>2</dailyProgress>
            </li>
      </progression>
      <triggers>
            <li>
                  <requirements>
                        <li
Class="RIMMSqol.remnantcolony.requirements.QOLCityRequirement_CrisisProgressIsAb
ove">
                              <crisis>MyCityStarvation</crisis>
                              <value>10.0</value>
                        </li>
                  </requirements>
                  <trigger
Class="RIMMSqol.remnantcolony.triggers.QOLCityTrigger_KillPopulation">
                        <numOfPeopleMax>1</numOfPeopleMax>
                        <numOfPeopleMin>1</numOfPeopleMin>
                        <hediff>MyCityDisease</hediff>
                        <chance>0.05</chance>
                  </trigger>
```

```
            </li>
      </triggers>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef ParentName="MyCityCrisis"
Class="RIMMSqol.remnantcolony.projects.QOLCityProjectDef_Crisis">
      <defName>MyCityOvertime</defName>
      <label>overtime</label>
      <description>A lack of recreation exposes people to health
risks.</description>
      <progression>
            <li>
                  <requirements>
                        <li
Class="RIMMSqol.remnantcolony.requirements.QOLCityRequirement_NeedValueIsNegativ
e">
                              <need>MyCityLabour</need>
                        </li>
                  </requirements>
                  <dailyProgress>2</dailyProgress>
            </li>
      </progression>
      <triggers>
            <li>
                  <requirements>
                        <li
Class="RIMMSqol.remnantcolony.requirements.QOLCityRequirement_CrisisProgressIsAb
ove">
                              <crisis>MyCityOvertime</crisis>
                              <value>10.0</value>
                        </li>
                  </requirements>
                  <trigger
Class="RIMMSqol.remnantcolony.triggers.QOLCityTrigger_KillPopulation">
                        <numOfPeopleMax>1</numOfPeopleMax>
                        <numOfPeopleMin>1</numOfPeopleMin>
                        <hediff>MyCityDisease</hediff>
                        <chance>0.05</chance>
                  </trigger>
            </li>
      </triggers>
</RIMMSqol.remnantcolony.QOLCityProjectDef>
```

The city needs some basic functionality. Trading with the city and exchanging colonists. Otherwise boosters wouldn't work and nobody could ever life in the city. Such functionality can be added by using projects that spawn automatically and then add actions. Alternatively they could be added as callbacks on resources, like the boosters. We also add an action that allows changing the settings on the city so that it can be renamed.

```
<RIMMSqol.remnantcolony.QOLCityDef>
      …
      <projects>
            …
            <li>MyCityCore</li>
      </projects>
</RIMMSqol.remnantcolony.QOLCityDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef>
      <defName>MyCityCore</defName>
      <label>city center</label>
      <description>Registers the basic actions each city has
available.</description>
```

```
      <order>-1</order>
      <lifetime>0</lifetime>
      <autospawnTimer>1</autospawnTimer>
      <cancellable>false</cancellable>
      <toggleable>false</toggleable>
      <labour>0</labour>
      <maximumInstances>1</maximumInstances>
      <sendCompletedNotification>false</sendCompletedNotification>
      <actions>
            <li>MyCityTradeWithCaller</li>
            <li>MyCityExchangePopulation</li>
            <li>MyCityCitySettings</li>
      </actions>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.actions.QOLCityActionDef>
      <defName>MyCityTradeWithCaller</defName>
      <label>trade</label>
      <description>Allows to trade resources between the city and a
caller.</description>
      <icon>Symbols/SkillSocial</icon>
      <order>1</order>
      <action
Class="RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_TradeWithCaller
" />
</RIMMSqol.remnantcolony.actions.QOLCityActionDef>

<RIMMSqol.remnantcolony.actions.QOLCityActionDef>
      <defName>MyCityExchangePopulation</defName>
      <label>exchange population</label>
      <description>Allows the exchange of population between a settlement and a
city.</description>
      <icon>Symbols/Population</icon>
      <order>2</order>
      <action
Class="RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_ExchangeCitizen
s" />
</RIMMSqol.remnantcolony.actions.QOLCityActionDef>

<RIMMSqol.remnantcolony.actions.QOLCityActionDef>
            <defName>MyCityCitySettings</defName>
            <label>settings</label>
            <description>Access to the city settings.</description>
            <icon>Symbols/Edit</icon>
            <order>100</order>
            <action
Class="RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_CitySettings" /
>
      </RIMMSqol.remnantcolony.actions.QOLCityActionDef>
```

If the city should be a victory condition then adding an achievement that ends the game after fulfilling certain requirements is possible. Here we say that growing the city to ten citizen is enough to win the game.

```
<RIMMSqol.remnantcolony.QOLCityDef>
      …
      <projects>
            …
            <li>MyCityAchievementGameWon</li>
      </projects>
</RIMMSqol.remnantcolony.QOLCityDef>

<RIMMSqol.remnantcolony.QOLCityProjectCategoryDef>
```

```
      <defName>MyCityAchievements</defName>
      <label>Achievements</label>
      <description>Acomplishments reached during the lifetime of the
city.</description>
      <order>30</order>

<renderer>RIMMSqol.remnantcolony.cities.QOLCityAchievementRenderer</renderer>
</RIMMSqol.remnantcolony.QOLCityProjectCategoryDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef Abstract="True"
Name="MyCityAchievement">
      <order>-1</order>
      <lifetime>0</lifetime>
      <autospawnTimer>1</autospawnTimer>
      <cancellable>false</cancellable>
      <labour>0</labour>
      <category>MyCityAchievements</category>
      <maximumInstances>1</maximumInstances>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.QOLCityProjectDef ParentName="MyCityAchievement">
      <defName>MyCityAchievementGameWon</defName>
      <label>victory</label>
      <description>You fulfilled the requirements for winning the
game.</description>
      <requirements>
            <li
Class="RIMMSqol.remnantcolony.requirements.QOLCityRequirement_PopulationCount">
                  <min>10</min>
                  <max>9999</max>
            </li>
      </requirements>
      <listeners>
            <li>MyCityTriggerVictory</li>
      </listeners>
</RIMMSqol.remnantcolony.QOLCityProjectDef>

<RIMMSqol.remnantcolony.listeners.QOLCityListenerDef
Class="RIMMSqol.remnantcolony.listeners.QOLCityListenerDef_TriggerVictory">
      <defName>MyCityTriggerVictory</defName>
      <label>victory</label>
      <description>This listener will trigger the game victory and deactivate
the project to prevent it from triggering multiple times.</description>
      <eventIds>
            <li>
                  <eventId>PostProjectBuild</eventId>
                  <order>1000</order>
            </li>
      </eventIds>

<listenerClass>RIMMSqol.remnantcolony.listeners.QOLCityListener_TriggerVictory</
listenerClass>
      <victoryMessage>QOLCityGameOverGeneric</victoryMessage>
</RIMMSqol.remnantcolony.listeners.QOLCityListenerDef>
```

Balancing tips

Methods described here are no absolute ruling, they are listed to help modders understand what problems they have to solve when creating their content and how they could be solved. If multiple modders are working on a joined project it is advisable to agree to a shared set of rules. That way the users won't see the seams in the patchwork.

Work and Labour

Work is the measure what the vanilla game uses. Labour is the measure used by cities. A single human can produce 60k work per day. After calculating the work necessary for sleeping, recreation and eating 35k are left. The amount of labour produced is arbitrary. It should be sufficiently small to allow spending it with single clicks and sufficiently big to allow measuring in natural numbers without excessive rounding. The demo city uses 5 labour.

When trying to determine how much a building should cost, use blueprints to layout the building with nominal vanilla structures. Then select the blueprints and sum up the shown work requirements. That number must be multiplied by 60(since rimworld divides it by 60 before displaying it) to become real work. Now you can divide work by 35k and multiply the result by the chosen labour per day to find out how much labour is needed to build the structure.

When using materials that have multiple choices choose one and stick with it. For example all stone structures use granite, all wool is alpaca wool and all leather is bearskin.

Wood structures reduce the work by a lot and if you offer wood and stone alternatives then penalizing wood is necessary to make stone relevant. That can be done by using a flammability/maintenance crisis or giving stone a higher wealth generation that attracts more people(good and bad).

Provided Implementations

Requirements

CrisisProgressIsAbove

RIMMSqol.remnantcolony.requirements.QOLCityRequirement_CrisisProgressIsAbove

Tests if the city has a RIMMSqol.remnantcolony.projects.QOLCityProject_Crisis project where the progress is greater or equal then the specified value. The crisis project must exist on the city.

Name	crisis
Type	RIMMSqol.remnantcolony.QOLCityProjectDef
Info	The configuration entry that is used by the crisis project instance.
Name	value
Type	float

Info	The value to compare with the crisis project's progress.

NeedValueIsNegative

RIMMSqol.remnantcolony.requirements.QOLCityRequirement_NeedValueIsNegative

Tests if the city has a specific need with a negative balance. If the need does not exist the requirement is not met.

Name	need
Type	RIMMSqol.remnantcolony.QOLCityNeedDef
Info	The need to test.

PopulationCount

RIMMSqol.remnantcolony.requirements.QOLCityRequirement_PopulationCount

Tests if the cities number of citizens is between min and max inclusive.

Name	min
Type	int
Info	The number of citizens must be at least this big.
Name	max
Type	int
Info	The number of citizens must be no bigger than this.

Or

RIMMSqol.remnantcolony.requirements.QOLCityRequirement_Or

Allows grouping multiple requirements. At least one of the grouped requirements must be valid for this requirement to be valid.

Name	requirements
Type	List<QOLCityRequirement>
Info	List of requirements of which at least one must be valid to have this requirement be valid.

And

RIMMSqol.remnantcolony.requirements.QOLCityRequirement_And

Allows grouping multiple requirements. All of the grouped requirements must be valid for this requirement to be valid.

Name	requirements
Type	List<QOLCityRequirement>
Info	List of requirements of which all must be valid to have this requirement be valid.

Not

RIMMSqol.remnantcolony.requirements.QOLCityRequirement_Not

Allows negating a requirement.

Name	requirement
Type	QOLCityRequirement
Info	Requirement to negate.

Chance

RIMMSqol.remnantcolony.requirements.QOLCityRequirement_Chance

Is valid if a random number between 0 and 1 inclusive is less or equal to the configured value.

Name	value
Type	float
Info	A value between 0 and 1 inclusive.

Pawn – Filter

Or

RIMMSqol.remnantcolony.pawnfilters.QOLCityPawnFilter_Or

Allows grouping multiple filter. At least one of the grouped filters must allow the request for this filter to allow the request.

Name	filters
Type	List<QOLCityPawnFilter>
Info	List of filters of which at least one must be allowed to have this filter allow the request.

And

RIMMSqol.remnantcolony.pawnfilters.QOLCityPawnFilter_And

Allows grouping multiple filter. All of the grouped filters must allow the request for this filter to allow the request.

Name	filters
Type	List<QOLCityPawnFilter>
Info	List of filters of which all must be allowed to have this filter allow the request.

Not

RIMMSqol.remnantcolony.pawnfilters.QOLCityPawnFilter_Not

Negates a filter.

Name	filter
Type	QOLCityPawnFilter
Info	Filter to negate.

Animal

RIMMSqol.remnantcolony.pawnfilters.QOLCityPawnFilter_Animal

Allows a request if the pawn's race properties define it as an animal(Not capable of using tools and flesh type is not mechanoid).

AtLeastHumanlikeIntelligence

RIMMSqol.remnantcolony.pawnfilters.QOLCityPawnFilter_AtLeastHumanlikeIntelligence

Allows a request if the pawn's race properties qualify as humanlike, which requires an intelligence of at least human level.

Flesh

RIMMSqol.remnantcolony.pawnfilters.QOLCityPawnFilter_Flesh

Allows a request if the pawn's race properties define a flesh type other than mechanoid.

PrisonerOfColony

RIMMSqol.remnantcolony.pawnfilters.QOLCityPawnFilter_PrisonerOfColony

Allows a request if the pawn is a prisoner of the player faction.

Actions

CitySettings

RIMMSqol.remnantcolony.actions.QOLCityAction_CitySettings

Displays a dialog in which the user can rename the city, delete the city or change the number of turns between city turns. Changing the frequency is intended for test purposes and defaults to 60k which equates to one day.

RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_CitySettings

Convert

RIMMSqol.remnantcolony.actions.QOLCityAction_Convert

Opens a dialog in which the user can set the maximum number of repetitions for a conversion. A conversion changes input contributions into output contributions. While the user can set a maximum number of repetitions, the actual repetitions might be lower based on available input contributions, required output contributions and labour.

RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_Convert

Name	inputPerCycle
Type	QOLCityStateContribution
Info	Which contributions will be removed for each cyle.
Name	outputPerCycle

Type	QOLCityStateContribution
Info	Which contributions are added for each cycle.
Name	labourPerCycle
Type	float
Info	How much labour is needed to perform each cycle.
Name	maxCyclePerInstance
Type	int
Info	How many cycles can be configured for each provided instance of this action.
Name	onlyNecessaryProduction
Type	bool
Info	Stops if the gains from the outputPerCycle are not needed.

RIMMSqol.remnantcolony.listeners.QOLCityListener_Conversion

The actual conversion takes place through the listener. A conversion is allowed if all needs and resources in the input contributions are positive after applying the contributions. A conversion is not necessary if all needs and resources in the output contributions are positive before applying the contributions.

Craft

RIMMSqol.remnantcolony.actions.QOLCityAction_Craft

Displays a dialog in which a work order can be configured and queued. Also the maximum amount of labour to use on pending orders.

RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_Craft

Name	workMultiplier
Type	float
Info	Amplifies the contributed labour into work that gets added to the recipe.
Name	maxLabourPerInstance
Type	int
Info	How much labour can be assigned for each provided instance of this action.
Name	factorWorkAwful, factorWorkPoor, factorWorkNormal, factorWorkGood, factorWorkExcellent, factorWorkMasterwork, factorWorkLegendary
Type	float
Info	Changes the base work amount for each quality level. Defaults to vanilla settings.
Name	categories
Type	List<ThingCategoryDef>
Info	Only recipes with all their products contained in these categories are usable.
Name	allowedQualityCategories
Type	List<QualityCategory>
Info	All quality categories that can be selected for the recipes, if they support quality levels.

RIMMSqol.remnantcolony.listeners.QOLCityListener_ActionCraft

Calculates how much labour is needed and available before allocating the resulting work to the queued orders. Once an order finished a notification is sent and the products appear in the cities storage.

ExchangeCitizens

RIMMSqol.remnantcolony.actions.QOLCityAction_ExchangeCitizens

Displays a dialog where the user can send people from the callers map into the city and vice versa. This includes any pawn belonging to the player faction or being imprisoned by the player faction. The actual list is modified by the immigration rules of the city. When citizens are send to the callers map they arrive via drop pod.

RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_ExchangeCitizens

Repair

RIMMSqol.remnantcolony.actions.QOLCityAction_Repair

Displays a dialog in which the user can set the maximum labour to be used for repairs.

RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_Repair

Name	categories
Type	List<ThingCategoryDef>
Info	All items in the cities storage that fit into one of these categories can be repaired.
Name	hitPointMultiplier
Type	float
Info	Amplifies labour spent to hitpoints repaired.
Name	maxLabourPerInstance
Type	int
Info	How much labour can be assigned for each provided instance of this action.

RIMMSqol.remnantcolony.listeners.QOLCityListener_ActionRepair

Repairs items that belong to any of the configured categories in the city storage. Items that have the deadman apparel flag and are at full hp will have it removed.

TradeWithCaller

RIMMSqol.remnantcolony.actions.QOLCityAction_TradeWithCaller

Opens the trade dialog to exchange items between the city and the callers settlement. The exchange takes place via orbital beacon/drops. All prices are incredibly low so that no silver is needed to exchange items.

RIMMSqol.remnantcolony.actions.QOLCityActionConfiguration_TradeWithCaller

Trigger

KillPopulation

RIMMSqol.remnantcolony.triggers.QOLCityTrigger_KillPopulation

When this trigger executes it has a percentage chance to kill a range of people in the city. The people are chosen at random.

Name	numOfPeopleMax
Type	int
Info	Maximum number of people that will be killed if this trigger executes.
Name	numOfPeopleMin
Type	int
Info	Minimum number of people that will be killed if this trigger executes.
Name	hediff
Type	HediffDef
Info	Optional cause of death.
Name	chance
Type	float
Info	Probability that this trigger tries to kill people. One equals 100% and 0 equals 0%.

Incident – Family Reunification

RIMMSqol.remnantcolony.triggers.QOLCityTrigger_IncidentFamilyReunification

If the player has a settlement it checks the citizens of the city and finds the ones with relatives that are not in the players faction, alive and not spawned. It then picks from those citizens a random one and spawns all those relatives on a players settlement. If any relative came from a hostile faction they will be chased by a raid.

If it was called through an event it marks the event as processed.

Callbacks

PeopleExchange

RIMMSqol.remnantcolony.callbacks.QOLCityCallback_PeopleExchange

Displays a dialog where the user can send people from the callers map into the city and vice versa. This includes any pawn belonging to the player faction or being imprisoned by the player faction. The actual list is modified by the immigration rules of the city. When citizens are send to the callers map they arrive via drop pod.

ResourceBooster

RIMMSqol.remnantcolony.callbacks.QOLCityCallback_ResourceBooster

Checks the storage of the city for required items. If available offers to convert them into a booster. If not it informs the user of missing resources. The conversion removes the required items from storage and adds a project to the city.

Name	thing
Type	ThingDef
Info	Which items are required.
Name	amount
Type	int
Info	How many items are required. Nutrition is currently not considered!
Name	boosterProject
Type	QOLCityProjectDef
Info	Which project to add to the city. It will build automatically if the labour requirement is 0 or less.

Trade

RIMMSqol.remnantcolony.callbacks.QOLCityCallback_Trade

Opens the trade dialog to exchange items between the city and the callers settlement. The exchange takes place via orbital beacon/drops. All prices are incredibly low so that no silver is needed to exchange items.

Projects

Crisis

RIMMSqol.remnantcolony.projects.QOLCityProject_Crisis

A project that has a progress variable which it advances based on its configuration through a listener. The progress should be displayed utilizing the configured display limits.

Name	progress
Type	float
Info	The crisis progress.

RIMMSqol.remnantcolony.projects.QOLCityProjectDef_Crisis

Name	dailyDecrease
Type	float
Info	By how much the progress recovers naturally. A negative value will turn the crisis into a doom clock.
Name	progression
Type	List<QOLCityProjectDef_Crisis_RequirementProgress>
Info	Contributors to the crisis progress that have prerequisites.

Name	triggers
Type	List<QOLCityProjectDef_Crisis_RequirementTrigger>
Info	A list of trigger that are executed if the prerequisites are met. There is no innate requirement for the crisis progress.
Name	maxProgression, minProgression
Type	float
Info	The boundaries for the crisis progress.
Name	maxDisplayProgression, minDisplayProgression
Type	float
Info	The boundaries for the displayed crisis progress. This allows to visualize a crisis hitting a critical point while also accumulating further progress and deepening the crisis.

RIMMSqol.remnantcolony.projects.QOLCityProjectDef_Crisis_RequirementProgress

Name	requirements
Type	List<QOLCityRequirement>
Info	If all requirements are fulfilled the daily progress is added to the crisis progression.
Name	dailyProgress
Type	float
Info	By how much the progress of the crisis is increased every day if the requirement is fulfilled.

RIMMSqol.remnantcolony.projects.QOLCityProjectDef_Crisis_RequirementTrigger

Name	requirements
Type	List<QOLCityRequirement>
Info	If all requirements are fulfilled the trigger is invoked when progressing the crisis.
Name	trigger
Type	QOLCityTrigger
Info	What to do when the requirements are met.

RIMMSqol.remnantcolony.listeners.QOLCityListener_CrisisProgression

Advances the crisis progress and executes trigger in accordance with their prerequisites. Crisis progression will be constricted to its limits after all progress is accounted for not on each individual step.

Cities

Default

RIMMSqol.remnantcolony.cities.QOLCity_Default

This implementation provides the GUI that displays all city elements and links their functionality. This implementation has the goal to support all standard functionality for any

number of configured objects and convey enough information to the user to play around with the city without reading an external document.

Actions can be clicked on and provide custom responses. Dialog, reports, etc.	Projects - Districts are out of town settlements that contribute to the city. - Buildings are built in town and contribute to the city. - Crisis are showing how far a situation has progressed and visualize urgency for addressing shortfalls in needs. - Achievements are pure vanity reminder of completed goals someone else has set for you.
Needs are listing what should be addressed to keep the city stable.	
Indicators give information about where your society is heading. Clicking an indicator opens a detail view.	

Table 3: Layout for default city

Configuration Field summary

Project

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	worldObject
Type	WorldObjectDef
Info	World object def that is used as a template to construct a world object representing a completed project in a city.
Name	order
Type	int
Info	Used to sort this def in relation to other instances of this def. Default order is from low to high.
Name	lifetime
Type	int
Info	A positive number above zero indicates that this project last for that number of turns before becoming inactive.
Name	autospawnTimer

Type int
Info A positive number indicates that this project is started after that number of turns. A negative number means it is started periodically after that number of turns until maxInstances are reached. Zero means it is build manually.

Name cancellable
Type bool
Info Whether an existing project can be removed by the user.

Name labour
Type float
Info How much labour is required to finish this project.

Name maximumInstances
Type int
Info How often can this project be repeated in one city.

Name requirements
Type List<QOLCityRequirement>
Info Conditions that must be met before the project can be added to the city. A requirement is a custom class with no default fields. The fields and their meaning are determined by the used class.

Name toggleable
Type bool
Info Whether the project can be manually activated/deactivated.

Name states
Type QOLCityStateContributions
Info Describes the influence the project has in its various states.

Name actions
Type List<QOLCityActionDef>
Info Lists all actions provided by each instance of this project for the owning city.

Name stateClass
Type Type
Info Can be provided to use a custom class when creating the state object for this configuration. If its null the default class will be used.

Name category
Type QOLCityProjectCategoryDef
Info How and where to display the project.

Name sendCompletedNotification
Type bool
Info Whether a notification is sent when the project was built.

Name listeners
Type List<QOLCityListenerDef>
Info Which listeners are to be registered on each instance of the def.

QOLCityStateContributions

Name	build
Type	QOLCityStateContribution
Info	Static contributions made to the city after this project has been built and is active.
Name	active
Type	QOLCityStateContribution
Info	Static contributions made to the city if this project has not been built but is active.
Name	inactive
Type	QOLCityStateContribution
Info	Static contributions made to the city if this project is deactivated.

QOLCityStateContribution

Name	indicators
Type	List<QOLCityStateContributionIndicator>
Info	An unsorted list of contributions to a cities society indicators.
Name	needs
Type	List<QOLCityStateContributionNeed>
Info	An unsorted list of contributions to a cities needs.
Name	resources
Type	List<QOLCityStateContributionResource>
Info	An unsorted list of contributions to a cities resources.

QOLCityStateContributionIndicator

Name	def
Type	QOLSocietyIndicatorDef
Info	Which society indicator will be modified.
Name	stageId
Type	string
Info	The id declared on a stage inside the QOLSocietyIndicatorDef.
Name	pawnFilters
Type	List<QOLCityPawnFilter>
Info	Restricts the selection of citizens that this contribution applies to.
Name	affinity
Type	float
Info	A value between 0 and 1 representing the percentage that the filtered citizens are aligned with the declared stage.
Name	suppression
Type	float

Info	A value between 0 and 1 representing by how much the filtered citizens affinity will be lowered for the declared stage.

QOLCityStateContributionNeed

Name	def
Type	QOLCityNeedDef
Info	Which city need will be modified. A need is split into provided and required contributions.
Name	provided
Type	NumericAdjustment
Info	Contribution to the available production.
Name	required
Type	NumericAdjustment
Info	Contribution to the requested production.

NumericAdjustment

Name	flat
Type	float
Info	An additive change to the final base value.
Name	factor
Type	float
Info	A percentile change to the base value, where 0 means it is unchanged and 0.2 would be a 20% increase. All contributing factors are summed up before a multiplication takes place.

QOLCityStateContributionResource

Name	def
Type	QOLCityResourceDef
Info	Which city resource will be modified.
Name	flat
Type	float
Info	An additive change to the final base value.
Name	factor
Type	float
Info	A percentile change to the base value, where 0 means it is unchanged and 0.2 would be a 20% increase. All contributing factors are summed up before a multiplication takes place.

QOLCityRequirement

A requirement is a custom class with no default fields. The fields and their meaning are determined by the used class.

QOLCityActionDef

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	icon
Type	string
Info	Path to an image or a folder. If it’s a folder the alphabetically last entry that is not ending with the mask prefix will be used.
Name	order
Type	int
Info	Used to sort this def in relation to other instances of this def. Default order is from low to high.
Name	action
Type	QOLCityActionConfiguration
Info	A requirement is a custom class with no default fields. The fields and their meaning are determined by the used class.
Name	listeners
Type	List<QOLCityListenerDef>
Info	Which listeners are to be registered on each instance of the def.

QOLCityProjectCategoryDef

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.

Name	order
Type	int
Info	Used to sort this def in relation to other instances of this def. Default order is from low to high.
Name	renderer
Type	Type
Info	A custom class that is extending “RIMMSqol.renderers.BaseRenderer” and with a public constructor that has exactly one parameter of type “Func<Flow,Pair<QOLCityProjectDef,List<QOLCityProject>>>”.

QOLCityListenerDef

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	listenerClass
Type	Type
Info	A custom class that is extending the abstract class “RIMMSqol.remnantcolony.listeners.QOLCityListener”. It must have a public constructor accepting exactly two parameters. First parameter must implement QOLCityListenerDef and will be an instance of this configuration object. The second parameter must implement IQOLCityListenerOwner and will be the object that owns this listener.
Name	eventIds
Type	List<EventProcessingProperties>
Info	A list of eventIds that are processed by this listener and their order in each eventId’s execution chain.

EventProcessingProperties

Name	order
Type	int
Info	Used to determine the order in which listener are processing occurring events.
Name	eventId
Type	string
Info	All events with a matching eventId will be processed by this listener.

QOLSocietyIndicatorDef

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	order
Type	int
Info	Used to sort this def in relation to other instances of this def. Default order is from low to high.
Name	stages
Type	List<QOLSocietyIndicatorStage>
Info	Defines numeric ranges with independent labels and contributions. The first range that includes the current value of the cities society indicator determines which stage is active.

QOLSocietyIndicatorStage

Name	id
Type	string
Info	An identifier that is unique amongst all stages of a QOLSocietyIndicatorDef.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	state
Type	QOLCityStateContribution
Info	Static contributions made to the city while this stage is active.
Name	thresholdMin
Type	float
Info	Lower border(inclusive) of the numeric range in which this stage is active.
Name	thresholdMax
Type	float
Info	Upper border(inclusive) of the numeric range in which this stage is active.
Name	listeners
Type	List<QOLCityListenerDef>
Info	Which listeners are to be registered on each instance of the def.

QOLCityNeedDef

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	order
Type	int
Info	Used to sort this def in relation to other instances of this def. Default order is from low to high.
Name	listeners
Type	List<QOLCityListenerDef>
Info	Which listeners are to be registered on each instance of the def.

QOLCityResourceDef

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	icon
Type	string
Info	Path to an image or a folder. If it’s a folder the alphabetically last entry that is not ending with the mask prefix will be used.
Name	callback
Type	QOLCityCallback
Info	Custom class implementing QOLCityCallback with custom configuration fields. The callback will be invoked when the user clicks the resource on the city screen.
Name	listeners
Type	List<QOLCityListenerDef>

Info	Which listeners are to be registered on each instance of the def.

QOLCityDef

Name	defName
Type	string
Info	A mandatory id that is unique among this type of definition. Can only contain letters, numbers or dashes. Can not be the text “null”.
Name	label
Type	string
Info	The text used to display the entry when viewed in the game.
Name	description
Type	string
Info	If it is defined it can not be empty. No leading or trailing spaces are allowed.
Name	worldObject
Type	WorldObjectDef
Info	The world object that holds the city state.
Name	projects
Type	List<QOLCityProjectDef>
Info	Which projects are available to this city.
Name	needs
Type	List<QOLCityNeedDef>
Info	Which needs are available to this city.
Name	indicators
Type	List<QOLSocietyIndicatorDef>
Info	Which indicators are available to this city.
Name	resources
Type	List<QOLCityResourceDef>
Info	Which resources are available to this city.
Name	labourDef
Type	QOLCityNeedDef
Info	Which of the cities needs represents labour.
Name	population
Type	QOLCityPopulationProperties
Info	Immigration policy and population contributions.
Name	traderKind
Type	TraderKindDef
Info	Allows to change what settlements are allowed to trade with the city.
Name	listeners
Type	List<QOLCityListenerDef>
Info	Which listeners are to be registered on each instance of the def.

QOLCityPopulationProperties

Name	socialStandings
Type	List<QOLCitySocialStandingDef>
Info	Used to assign citizens weight for the purpose of calculating contributions.
Name	immigrationLaws
Type	List<QOLCityPawnFilter>
Info	Custom classes that determine if a pawn can be added to the cities population.
Name	contributions
Type	List<QOLCityPopulationContribution>
Info	All contributions made by any group of citizens.

QOLCityPopulationContribution

Name	pawnFilters
Type	List<QOLCityPawnFilter>
Info	Custom classes deciding if a citizen makes this contribution or not.
Name	multiplierPopulationCount
Type	float
Info	Amplifies the number of valid citizens when calculating contributions. One means no change.