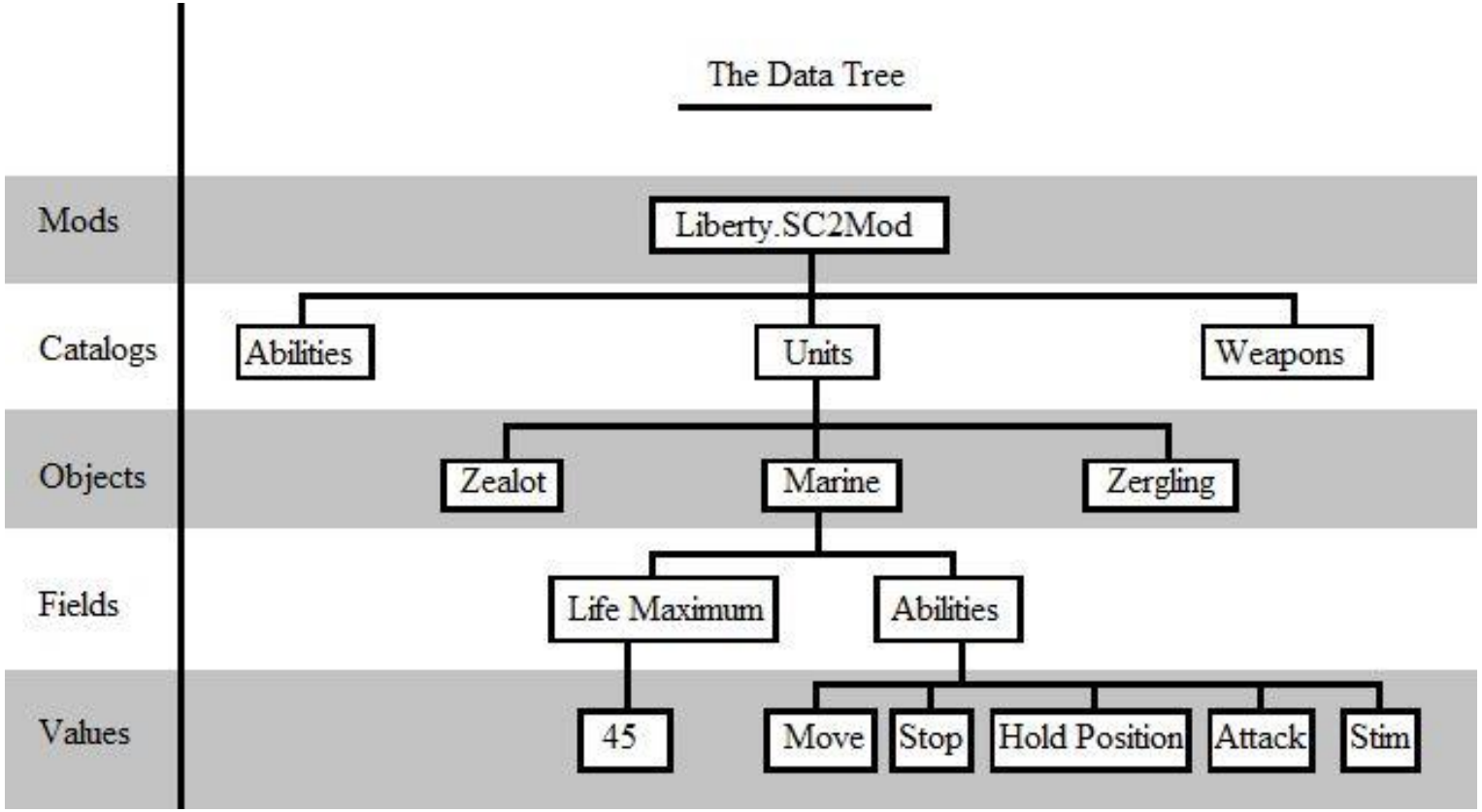
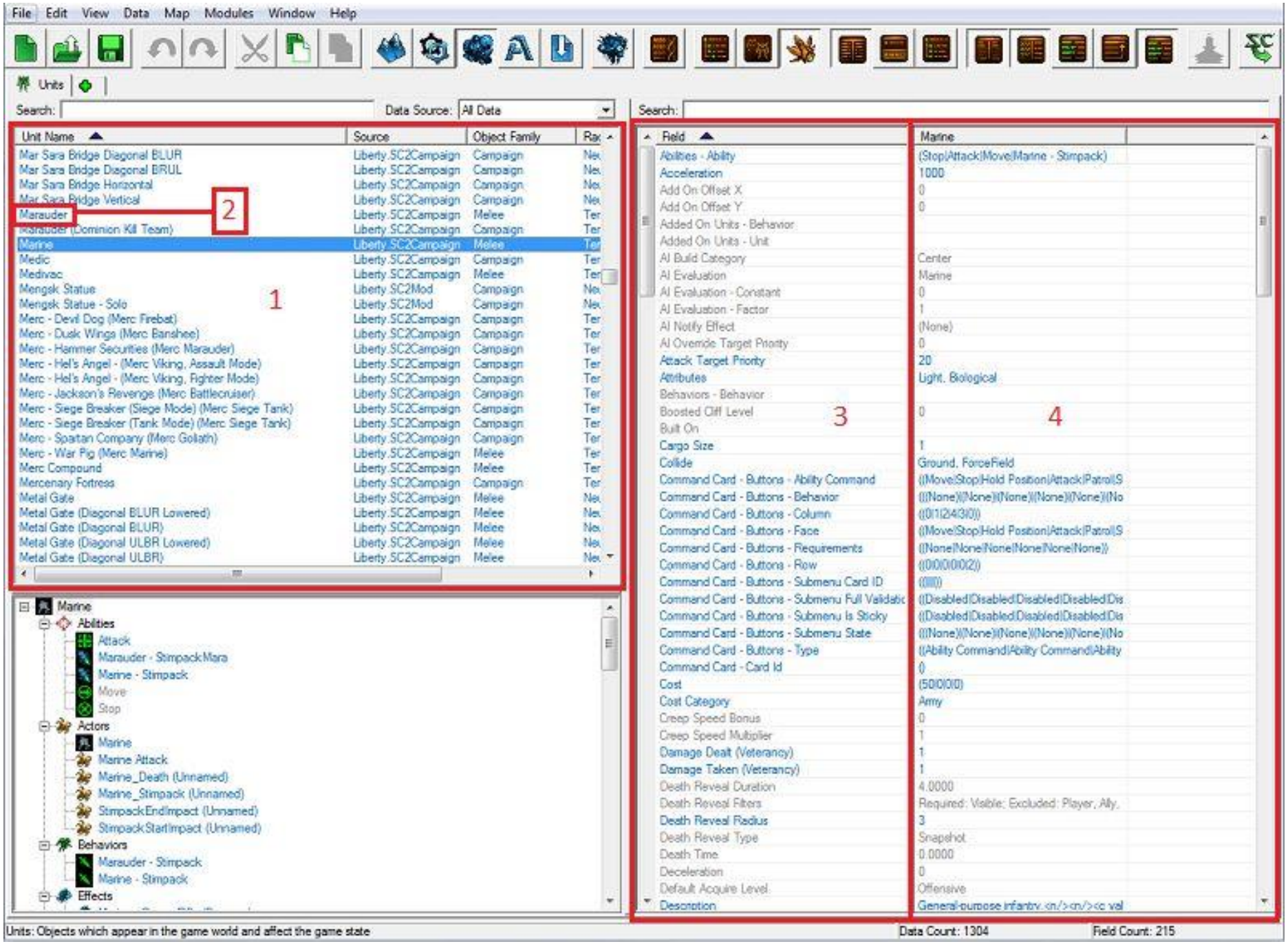
策划配置系统

Agenda

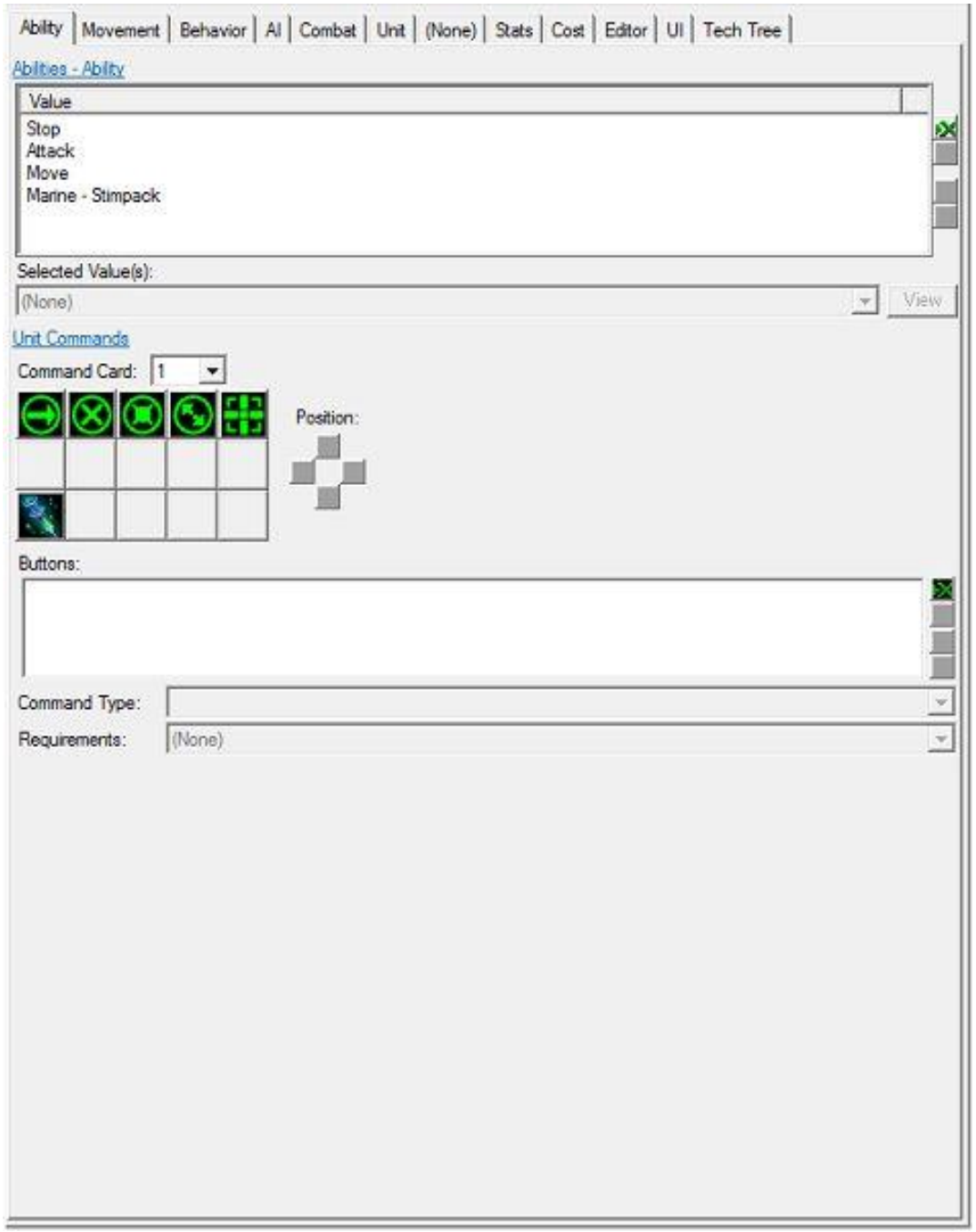
- 《星际争霸2》《极度恐慌》《神秘海域》的策划配置
- Configgen功能介绍
  - 8个方面
- 生成java，lua代码
  - Java 关注安全
  - Lua 关注启动速度和内存
- Configgen大致实现

星际争霸2-Data Module

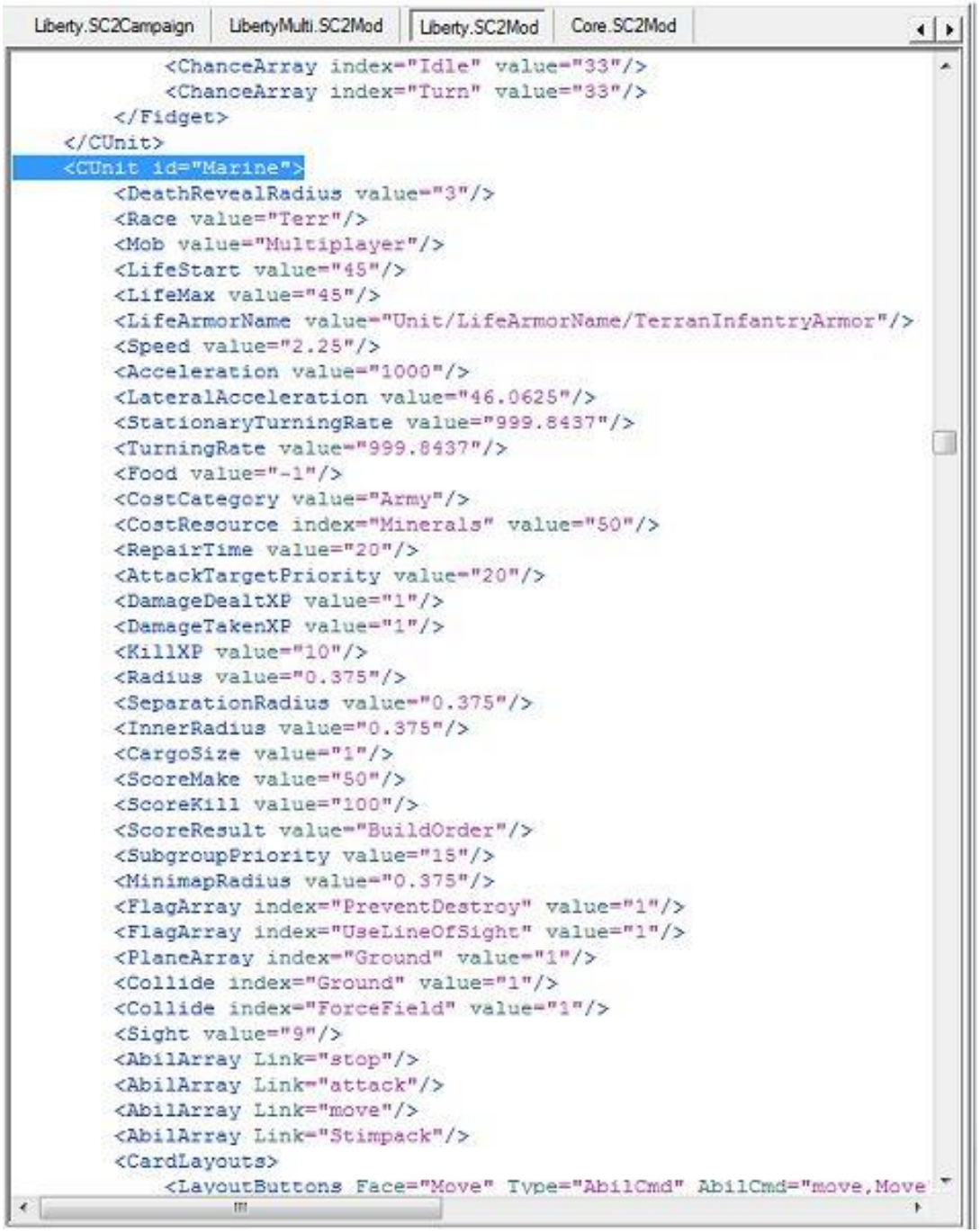




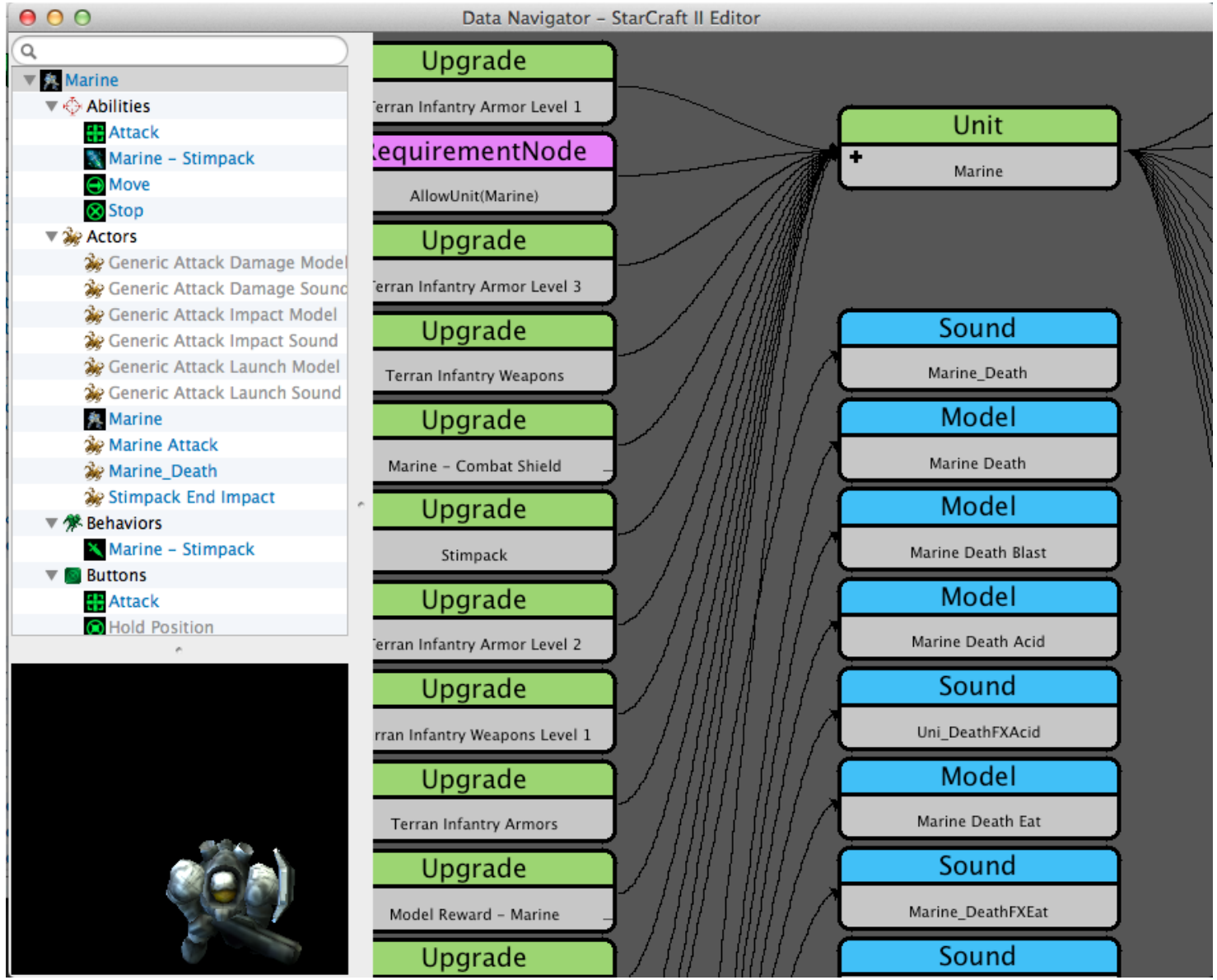
1. Catalog
2. Object
3. Field
4. Value



Detail View

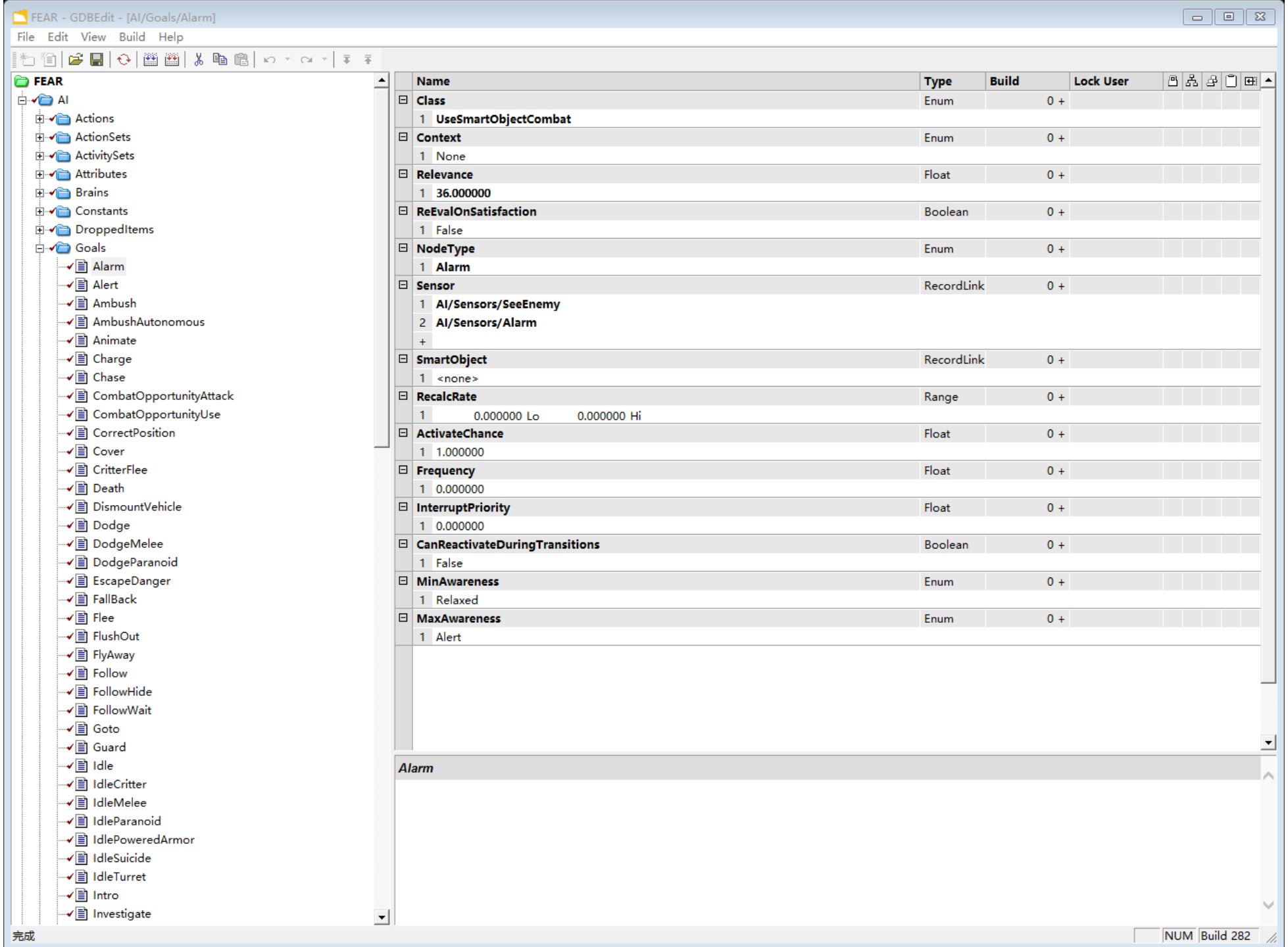


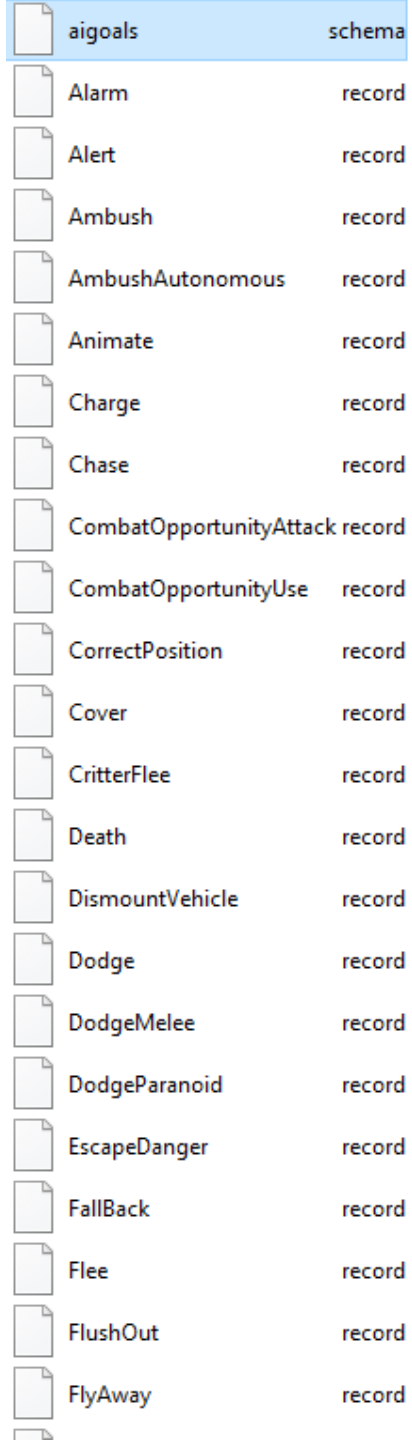
XML View [UnitData.xml]


Data Navigator – StarCraft II Editor
Marine
Abilities
Attack
Marine – Stimpack
Move
Stop
Actors
Generic Attack Damage Model
Generic Attack Damage Sound
Generic Attack Impact Model
Generic Attack Impact Sound
Generic Attack Launch Model
Generic Attack Launch Sound
Marine
Marine Attack
Marine_Death
Stimpack End Impact
Behaviors
Marine – Stimpack
Buttons
Attack
Hold Position
Upgrade
Terran Infantry Armor Level 1
RequirementNode
AllowUnit(Marine)
Upgrade
Terran Infantry Armor Level 3
Upgrade
Terran Infantry Weapons
Upgrade
Marine – Combat Shield
Upgrade
Stimpack
Upgrade
Terran Infantry Armor Level 2
Upgrade
Terran Infantry Weapons Level 1
Upgrade
Terran Infantry Armors
Upgrade
Model Reward – Marine
Upgrade
Unit
Marine
Sound
Marine_Death
Model
Marine Death
Model
Marine Death Blast
Model
Marine Death Acid
Sound
Uni_DeathFXAcid
Model
Marine Death Eat
Sound
Marine_DeathFXEat
Sound

极度恐慌 FEAR

Game Database





文件

Schema

```
aigoals.schema
[Schema]
Name=AI.Goals
Parent=
Help=Defines an AI Goal record.

[Attrib.Class]
Type=Enum
Data=..\aigoals.enum
Default=None
Values=1
Help=AIGoal Class (C++ class in code) that this Goal Type is base

[Attrib.Context]
Type=Enum
Data=..\aicontexts.enum
Default=None
Values=1
Help=Optional AI Context that this Goal sets on activation.

[Attrib.Relevance]
Type=Float
Data=min=0.0
Default=0.0
Values=1
Help=Static intrinsic relevance value.

[Attrib.ReEvalOnSatisfaction]
Type=Boolean
Default=False
Values=1
Help=TRUE if goal should recalculate relevance when it is satisfi

[Attrib.NodeType]
Type=Enum
Data=..\ainodes.enum
Default=None
Values=1
Help=Type of AINode.

[Attrib.Sensor]
Type=RecordLink
Data=Public AI.Sensors
Values=-1
Help=Required sensors for this goal.

[Attrib.SmartObject]
Type=RecordLink
Data=Public AI.SmartObjects
Default=<none>
Unicode=False
Deleted=False
```

Record

```
Alarm.record
[Record]
Schema=AI.Goals
Name=Alarm
Comment=
VirtualRelativeCategory=

[Attrib.Class]
Inherit=False
PlaceHolder=False
Todo=False
Modified=False
Lock=
Comment=
Value.0000=UseSmartObjectCombat

[Attrib.Relevance]
Inherit=False
PlaceHolder=False
Todo=False
Modified=False
Lock=
Comment=
Value.0000=36.000000

[Attrib.ReEvalOnSatisfaction]
Inherit=False
PlaceHolder=False
Todo=False
Modified=False
Lock=
Comment=
Value.0000=False

[Attrib.NodeType]
Inherit=False
PlaceHolder=False
Todo=False
Modified=False
Lock=
Comment=
Value.0000=Alarm

[Attrib.Sensor]
Inherit=False
PlaceHolder=False
Todo=False
Modified=False
Lock=
Comment=
Value.0000=AI/Sensors/SeeEnemy
Value.0001=AI/Sensors/Alarm
```

总结

- 以Record为单位来配置，Field都可配置默认值
- 星际用的.xml可以是任意复杂组合的结构，而FEAR用的.ini在基础平面结构上添加了列表.数组的支持
- 都有schema的定义，编辑器则都或多或少由schema自动生成
- 都提供了record之间关联的功能

神秘海域-Data Compiler

```
(define-state-script ("falling-sign")
  (state ("untouched")
    (on (update)
      [when [task-complete? "wz-post-combat"]
        [go "fallen"]
      ]
    )
    (on (event "hanging-from")
      [go "breaking"]
    )
  )
  ...
  (state ("breaking")
    (on (begin)
      [spawn-particles-at-joint "self"
        "hinge"
        "sign-break-dust"]
      [wait-animate "self" "sign-break"]
      [go "fallen"]
    )
  )
  (state ("fallen")
    (on (begin)
      [animate "self" "sign-broken"] ;; looping
    )
  )
)
```

```
(new ai-weapon-skill
    ;；weapon定义art，FX，sounds。type定义使用哪些Animation
    :weapon 'assault-rifle-b
    :type (ai-weapon-anim-type machine-gun)

    ;；伤害，分为对人和对物
    ;; damage parms
    :character-shot-damage 5
    :object-shot-damage 120

    ;；定义射击pattern，射击间隔，每次射几发子弹
    ;; rate of fire parms
    :initial-sequence-delay (rangeval 0.0 0.0)
    :num-bursts-per-sequence (rangeval-int 10000 10000)

    :auto-burst-delay (rangeval 0.4 0.8)
    :auto-burst-shot-count (rangeval-int 3 5)

    :single-burst-chance 0.33
    :single-burst-delay (rangeval 0.4 0.8)
    :single-burst-shot-count (rangeval-int 1 3)
    :single-burst-fire-rate (rangeval 0.16 0.20)

    ;；根据距离来调整准确度，越近越准确
    ;; accuracy parms
    :accuracy-curve *accuracy-assault-rifle-upgrade*

    ;；玩家从掩体探出头后的3.5秒内，这个枪的准确度从0，慢慢恢复到正常
    :time-to-accurate-cover 3.5
    :accuracy-cover-best 1.0
    :accuracy-cover-worst 0.0
```

总结2

- 星际由xml和galaxy脚本构成
- FEAR则不用脚本
- 神秘海域则都是lisp脚本，数据在脚本里

策划配置

- Script 脚本
- Xml 树结构，手工配置或工具生成
- Csv 表格结构

1：每列定义程序用名和类型

- 自动生成加载代码
- 策划可以检验所填数据是否符合类型

```
<column desc="脱战范围" name="PathRadius" type="int"/>
<column desc="警戒范围" name="AlertRadius" type="int"/>
<column desc="是否随机游荡" name="IsRanbdMove" type="int"/>
<column desc="随机游荡范围" name="RandMoveDis" type="int"/>
```

- 支持list，map（武林1/10是list，1/150是map）

```
<column compress="#" desc="女性奖励" name="woman" own="client" type="list,Drop"/>
    ref="item.commonitem" type="list,int"/>
<column desc="武器追加特效颜色" name="extraColors" own="client" type="list,string,6"/>
<column desc="升级概率，总和为10000" name="RateForUpgrade_List" type="list,int,6"/>
    ref="equip.runefightattr" type="list,int,4"/>

   <column name="attrmodifymap" own="effecteditor" type="map,int,buff.ModifyByTime,3"/>
<column desc="同福闯天关" name="Aid2NormalExpMap" own="client" type="map,int,int,9"/>
<column desc="颜色id" name="colorId2hsvGroupMap" own="client" type="map,float,HSVGroup,8"/>
```

2：每张表有主键和唯一键

- 生成根据键值来找到行的代码，提高了查找效率

```
<table name="npc.monster" own="effecteditor,client" primaryKey="MonsterID">
```

```
public static Monster get(int monsterID) {
    config.ConfigMgr mgr = config.ConfigMgr.getMgr();
    return mgr.npc_monster_All.get(monsterID);
}
```

- 支持主键有多个Key（武林有1/15的table使用）

```
<table name="shenbing.cost" own="client" primaryKey="ShenBingID,ShenBingLvl">
<table name="shenbing.model" own="client" primaryKey="ShenBingID,ShenBingLevel">
<table name="shenbing.randgroup" own="client" primaryKey="RandGroup,Luck">
<table name="skill.skillext.pretageskills" own="client" primaryKey="ID,Level">
<table name="skill.skillext.skillcostbook" own="client" primaryKey="id,level">
<table name="skill.skilllevel" own="effecteditor" primaryKey="SkillId,SkillLevel">
```

3：可配置表之间的索引关系

- 自动生成ref，nullableRef代码
- 策划可检验所填数据是否符合这个索引约束

```
<column desc="对应CharModel的Id" name="CharModelId" own="client"
    ref="model.charmodel" type="int"/>
<column desc="普攻技能Id" name="NormalSkillID" ref="skill.skill" type="int"/>
<column desc="技能1ID" name="SkillIDList" ref="skill.skill" type="list,int,4"/>
```

```
/**
 * 普攻技能Id
 */
public int getNormalSkillID() { return normalSkillID; }

public config.skill.Skill refNormalSkillID() { return RefNormalSkillID; }

/**
 * 技能1ID
 */
public java.util.List<Integer> getSkillIDList() { return skillIDList; }

public java.util.List<config.skill.Skill> refSkillIDList() { return RefSkillIDList; }
```

3.1：可配置索引到另一表多行，listRef

- 策划可以把一个列很多的表拆成2个，变成多行来减少列数
- 对程序来说，代码不变

```
<foreignKey keys="Id" name="Attr" ref="inspiration.attr,Type" refType="LIST"/>
```

```
public java.util.List<config.inspiration.Attr> listRefAttr() {
    return ListRefAttr;
}
```

```
ref="equip.equipset,SetId" refType="LIST" type="int"/>
ref="lifeplate.shufflequality,QualityGroup" refType="LIST" type="int"/>
ref="lifeplate.shuffleattr,ShuffleSlot" refType="LIST" type="int"/>
 ref="pet.petcatchpool,Id" refType="LIST" type="int"/>
```

4：类型支持自定义结构，多态结构

- 策划可更灵活的配置，同时兼具类型检测，索引检测这些功能

```
<bean enumRef="task.completeconditiontype"
    name="task.completecondition" own="client">
    <bean name="KillMonster">
        <column name="monsterid" ref="npc.monster" type="int"/>
        <column name="count" type="int"/>
        <column name="pointid" type="string"/>
    </bean>
    <bean name="TalkNpc">
        <column name="npcid" ref="npc.npc" type="int"/>
        <column name="talkid" ref="task.npctalk" type="int"/>
        <column name="targetpoint" type="string"/>
    </bean>
    <bean name="CollectItem">
```

R	S	T	
完成条件详见(	完成参数1	完成参数2	完成
completeconc	P1	P2	P3
TalkNpc	13034	1	
WorldShout	嘿兄弟我们好久不见		
TalkNpc	13034	131002	
FindNpc	13035		
FindNpc	13035		
FindNpc	13035		
TalkNpc	13035	111007	
MirrorKillMons	10083		
TalkNpc	13035	111008	
TalkNpc	13559	13100850	
ReachPoint	zc01qixiazhen_task_lujianbuping		

武林的Bean

- 概念上有个合理的分层
- 增加了跨表的一致性
- 方便程序统一处理

```
<bean compress=";" name="Drop" own="client">
    <column desc="物品编号" name="itemid" own="client"
        ref="item.commonitem" type="int"/>
    <column desc="数量" name="num" own="client" type="int"/>
</bean>
<bean name="DropItemInfo">
    <column desc="掉落类型" name="droptype" type="int"/>
    <column desc="物品" name="itemid" range="1,2000000000" type="int"/>
    <column desc="掉落概率" name="droprate" type="float"/>
    <column desc="数量" name="count" range="1,200000" type="int"/>
    <column desc="是否珍稀物品" name="isprecious" type="int"/>
</bean>
<bean name="DropItemInfoForShow" own="client">
    <column desc="掉落类型" name="droptype" own="client" type="int"/>
    <column desc="物品" name="itemid" own="client"
        range="1,2000000000" type="int"/>
    <column desc="数量" name="count" own="client" range="1,200000" type="int"/>
</bean>
<bean name="DropList" own="client">
    <column compress="#" desc="普通道具奖励" name="common" own="client" type="list,Drop"/>
    <column compress="#" desc="剑客道具奖励" name="JianKe" own="client" type="list,Drop"/>
    <column compress="#" desc="枪豪道具奖励" name="QiangHao" own="client" type="list,Drop"/>
    <column compress="#" desc="医师道具奖励" name="YiShi" own="client" type="list,Drop"/>
    <column compress="#" desc="术士道具奖励" name="ShuShi" own="client" type="list,Drop"/>
    <column compress="#" desc="神机" name="ShenJi" own="client" type="list,Drop"/>
    <column compress="#" desc="扩展职业1" name="ExtProfession1"
        own="client" type="list,Drop"/>
    <column compress="#" desc="扩展职业2" name="ExtProfession2"
        own="client" type="list,Drop"/>
    <column compress="#" desc="扩展职业3" name="ExtProfession3"
        own="client" type="list,Drop"/>
    <column compress="#" desc="男性奖励" name="man" own="client" type="list,Drop"/>
    <column compress="#" desc="女性奖励" name="woman" own="client" type="list,Drop"/>
    <column compress=";" desc="称号道具" name="TitleList" own="client"
        ref="item.commonitem" type="list,int"/>
</bean>
```

武林的多态bean

<bean name="achievement.completecondition">
<bean name="FinishAchievement">
<bean name="FinishTask">
<bean name="FinishActivity">
<bean name="RoleLevel">
<bean name="HaveFriends">

<bean name="task.completecondition"
<bean name="KillMonster">
<bean name="TalkNpc">
<bean name="CollectItem">
<bean name="UseItem">
<bean name="ReachPoint">
<bean name="ReachPoints">
<bean name="Dye">
<bean name="GuardNpc">
<bean name="ReachLevel">

<bean name="buff.BuffLogic" ">
<bean name="Damage">
<bean name="DamageModifier">
<bean name="DamageImmune">
<bean name="DamageAbsorbReflect">
<bean name="Heal">
<bean name="HealModifier">
<bean name="Period">
<bean name="AttrModifier">
<bean name="God">
<bean name="Immune">
<bean name="Dispel">
<bean name="Mark">

<bean name="task.unlock">
<bean name="Level">
<bean name="TaskAndLevel">
<bean name="AddCoin">
<bean name="Star">
<bean name="Family">

5：单元格可写任意复合结构数据

- 灵活性进一步增加，等价于xml的表达能力了

```
<bean enumRef="task.completeconditiontype" name="task.completecondition">
    <bean name="KillMonster">
        <column name="monsterid" ref="monster" type="int"/>
        <column name="count" type="int"/>
    </bean>
    <bean name="TalkNpc">
        <column name="npcid" type="int"/>
    </bean>
    <bean name="Chat">
        <column name="msg" type="string"/>
    </bean>
    <bean name="ConditionAnd">
        <column compressAsOne="1" name="cond1" type="task.completecondition"/>
        <column compressAsOne="1" name="cond2" type="task.completecondition"/>
    </bean>
    <bean name="CollectItem">
        <column name="itemid" type="int"/>
        <column name="count" type="int"/>
    </bean>
</bean>
```

程序用名字			
name	completecondition	param1	param2
杀个怪	KillMonster	1	3
和npc对话	TalkNpc	1	
收集物品	CollectItem	11	1
杀怪并且收集物品	ConditionAnd	KillMonster(1,3)	CollectItem(11,1)
杀怪对话并且收集物品	ConditionAnd	ConditionAnd(KillMonster(1,3), TalkNpc(1))	CollectItem(11,1)
聊天并且杀怪	ConditionAnd	Chat("葵花宝典,123")	KillMonster(1,3)

在excel一格中配置复杂结构

- 武林中有1/20的column用的单格压缩

- 旧的方案：在bean上，在column上配置compress，使用不同的分割符来构造复杂结构

普通道具奖励

DropList@Common

518;4#511;1600

518;4#511;1608

518;4#511;1616

518;4#511;1624

```
<bean compress=";" name="Drop" own="client">
    <column desc="物品编号" name="itemid" own="client"
            ref="item.commonitem" type="int"/>
    <column desc="数量" name="num" own="client" type="int"/>
</bean>

<bean name="DropList" own="client">
    <column compress="#" desc="普通道具奖励" name="common" own="client" type="list,Drop"/>
```

缺点

- 嵌套的bean，list必须使用不同的分隔符
- 没法表示带递归的多态结构
  - 比如：ConditionAnd(cond1: Condition, cond2: Condition)
- 在bean上定义compress后所有用到这个bean的table都必须compress了。感觉compress不该是bean的属性，而应该是column的属性

解决方案： 用()构成层级树结构

- 只在column上配置compressAsOne就好，不用再配置分隔符，都用逗号来分割

- 里面的bean,list用()扩起来，如果bean是多态，则()左边加上子bean名称

```
    <column compressAsOne="1" desc="测试compressAsOne" name="TestCompress1" type="list,TestCompressBean"/>
</table>

<bean name="TestCompressBean">
    <column name="name" type="string"/>
    <column name="range" type="Range"/>
</bean>
```

测试compressAsOne
TestCompress1
("range1",(100,120)),("range2", (300,400))

6：支持定义枚举-避免魔数

- 程序可直接根据name找到表中一行，不需要根据magic number
- 分enum和enumPart，enum不支持服务器热更

```
<table enum="name" name="buff.bufflogictype"
    own="effecteditor,client" primaryKey="id">
    <column desc="id" name="id" type="int"/>
    <column desc="name" name="name" type="string"/>
    <column desc="是否可叠加（用于新的Buff叠加机制
        name="canSuperposition" type="bool"/>
```

```
<table enumPart="Ename" name="map.copyscene"
    own="effecteditor,client" primaryKey="Id">
    <column name="Id" own="client" type="int"/>
    <column desc="镜像副本" name="isMirror" own="client" type="bool"/>
    <column desc="跨服类型，参见crosstype.csv" name="CrossServerType"
        own="client" ref="map.crosstype" type="int"/>
    <column desc="父活动ID六扇门练血堂有父" name="ParentActivityId"
        ref="map.copyscene" refType="NULLABLE" type="int"/>
    <column desc="程序用名" name="Ename" own="client" type="string"/>
```

id	name	是否可叠加（用于
id	name	canSuperposition
1	Damage	
2	Heal	
3	Period	
4	AttrModifie	TRUE
5	DamageImmune	
6	Immune	
7	Mark	TRUE
8	Stun	
9	Sleep	

武林配置中的枚举

id	name	是否可叠加	优先级（动	动作（没有	特效（引用	显示进度条	移动限制	施法限制	策划说明	参数说明p1	p2	p3	p4
id	name	canSuperpo	priority	anim	sfx	hasbar	movedisab	skilldisabled					
1	Damage								伤害，共18个参数，按	伤害百分比修正(0表示不造成伤害，	伤害百分比	根据类型增加伤	
2	Heal								治疗	治疗百分比(0表示不造成伤害，100	附加攻击值	hp回复绝对	hp回
3	Period								持续	间隔时间毫秒	buff列表，分号分割。		
4	AttrModifier	TRUE							战斗属性修改	属性1id，参考config.common.fight	属性1绝对	属性1百分	属性
5	DamageImmune								掉血免疫	最小伤害	条件（Trig	条件参数1	条件
6	Immune								免疫	可免疫buff类别	免疫buff	免疫最大数量	
7	Mark	TRUE							标记,暂时没用。某些技	组Id，表示一组标记逻辑，引用buf	排序Id，数	值越小越先	
8	Stun		5	stun	buff_sfx_xu	TRUE	TRUE	TRUE	眩晕	是否必中			
9	Sleep		4	stun	buff_sfx_hu	TRUE	TRUE	TRUE	昏睡	是否必中			
10	Silent		3		buff_sfx_ch	TRUE		TRUE	沉默	是否必中			
11	Root		2		buff_sfx_ch	TRUE	TRUE		缠绕	是否必中			
12	Interrupt								打断当前技能	无			
13	Taunt								嘲讽	无			
14	Dispel								驱散	可驱散buff类别,subclass	可驱散buff	最大驱散数量	

全枚举的生成代码

```
public enum Rank {
    WHITE( name: "white",  value: 1),
    GREEN( name: "green",  value: 2),
    BLUE( name: "blue",  value: 3),
    PURPLE( name: "purple",  value: 4),
    YELLOW( name: "yellow",  value: 5);

    public config.equip.Rank_Detail ref() { return config.equip.Rank_Detail.get(value); }
```

```
public class Rank_Detail {
    private int rankID;
    private String rankName;
    private String rankShowName;
```

半枚举的生成代码

```
public class Jewelrysuit_Entry {
    public static Jewelrysuit_Entry SPECIALSUIT = new Jewelrysuit_Entry( name: "SpecialSuit",  value: 4);

    public config.equip.Jewelrysuit ref() { return config.equip.Jewelrysuit.get(value); }
```

```
public class Jewelrysuit {
    private int suitID;
    private String ename;
    private String name;
    private int ability1;
    private int ability1Value;
    private int ability2;
```

7：更多的约束检测，以及部分提取

- 策划可配置数字，字符串长度限制range，这样如果配错打表不通过
- CfgCheck.java这是个扩展点，策划可提要求来让程序支持更多约束检测，这样如果配错服务器不启动

```
<column desc="伪随机掉落周期（次数N的值，周期的意思根据宝箱类型）" name="FakeCycle"
    range="1,1000" type="int"/>
```

- 通过own配置可以以column为单位提取部分配置数据，节省客户端内存
- 也支持通过import配置，分serverconfig.xml, clientconfig.xml，和诛仙一样根据目录来单独生成部分数据。

8：国际化支持

- 支持国际化到单语言
- 支持多国语言客户端切换，并登陆同一个服务器

- 类型从string改为text
- 多国语言支持，需要服务器记录下此用户的所用语言，然后发送比如公告协议时，自动选择对应语言。（text字段配置上保存这对应的多国语言，gm平台工具也得支持多国语言）

列数太多怎么办?

- 一个复杂的树结构 如何拆分到多个表，使列数不太多?
  - 切割bean，把一个bean分成多个，共享相同的primary key，通过此key相连
  - 切割bean里的list<bean2>，每个bean2在另一个table中，bean2里配置一个column为bean的primary key，然后在bean的primary key上配置ref, refType="LIST"，与bean2相连

武林配置的schema统计（20200428）

- 706个Bean
  - 220个正常Bean
  - 23个多态Bean – 差不多500个子Bean

- 1429个Table（全部都有Primary Key）
  - 92个多列为主键
  - 217个全枚举，Enum
  - 162个半枚举，EnumPart
  - 8个Unique Key

- 9504个Column
  - 2431个外键约束，Foreign Key，ref
    - 527个是NULLABLE
    - 15个是LIST
  - 930个是list，61个map
  - 512个配置了只占excel一格，compress（之后都用compressAsOne）
  - 9个配置了取值范围约束，Range

生成代码

- 会直接生成Record Link的访问代码
  - refXxx
  - nullableRefXxx
  - listRefXxx

- 对配置的查询比如config.item.Commonitem.get(itemId)，只应存在在协议的处理函数啊这些入口，内部的函数里尽量不要有配置查询，因为这些link访问代码已经生成了

生成java--关注安全

- 读取配置分2阶段
  - Read 读取配置数据
  - Resolve 外键，检验数据约束
    - 如果refXxx，使用时可不用判null（因为resolve阶段会抛异常），如果是nullableRef则要判null

生成java

- 所有的配置有一个统一的入口
  - reload时多线程会正确一点

```
public class ConfigMgr {
    private static volatile ConfigMgr mgr;

    public static ConfigMgr getMgr() { return mgr; }

    public static void setMgr(ConfigMgr newMgr) { mgr = newMgr; }

    public final java.util.Map<Integer, config.equip.Jewelry> equip_jewelry_All = new java.util.LinkedHashMap<>();

    public final java.util.Map<config.LevelRank, config.equip.Jewelryrandom> equip_jewelryrandom_All = new java.util.LinkedHashMap<>();
```

生成java

- 生成的java代码会附带schema（ConfigCodeSchema.java）
  - 1，不用从生成的的访问类反射取结构
  - 2，把枚举的具体值视为schema

```
private static Schema equip_rank() {
    SchemaEnum s2 = new SchemaEnum(false, true);
    s2.addValue("white", 1);
    s2.addValue("green", 2);
    s2.addValue("blue", 3);
    s2.addValue("purple", 4);
    s2.addValue("yellow", 5);
    return s2;
}

private static Schema equip_rank_Detail() {
    SchemaBean s2 = new SchemaBean(true);
    s2.addColumn("RankID", SchemaPrimitive.SInt);
    s2.addColumn("RankName", SchemaPrimitive.SStr);
    s2.addColumn("RankShowName", SchemaPrimitive.SStr);
    return s2;
}
```

生成java

- 生成的java二进制数据在数据头自带schema

- Reload时会比较代码code的schema是否和数据data的schema兼容

```
public interface Schema {
    boolean compatible(Schema other);
```

- 兼容给才能reload成功
  - 全枚举必须不变，但枚举里的具体属性可变
  - 半枚举只能增，不能减
  - 可增加table，不能减少table

生成lua—关注加载速度和内存最小化

```lua
local this = cfg.equip.customequip

local mk = cfg._mk.table(this, uniqkeys: { { "all", "get", 1 }, }, enumidx: nil, refs: {
    { "RefId", false, cfg.item.commonitem, "get", 1 },
    { "RefEquipAttr", false, cfg.equip.equipattr, "get", 3 }, },
    "id", -- int, #ID
    "name", -- string, 名称
    "equipAttr", -- int, 装备属性
    "refineLevel"  -- int, 精炼等级
    )

mk(120015, "枪豪", 61108, 8)
mk(120016, "剑客", 71108, 8)
mk(120017, "术士", 41108, 8)
mk(120018, "医师", 51108, 8)
mk(120028, "神机", 161108, 8)
mk(120029, "花雨", 171108, 8)
mk(120030, "暗影", 181108, 8)

return this
```

lua优化

- 最初是二进制存储然后再zip,启动时lua读取zip加载
- 上线前优化：数据存储到lua文件里,table用array。
  - 加载速度提高10倍，从2.3s提高到0.24s,
- 把数据格式从{a=1,b=2,c=3}变成{1,2,3}(利用metatable,__index程序接口不变),
  - 内存从42M优化到26M。

Lua优化

- 延迟加载lua,只有需要用到表的时候再require。比如调用cfg.item.commonitem.get(xxx)调用到.get时再去require "cfg.item.commonitem"表。
  - 可以提升加载速度和减少内存占用

- 2019年12月(上线1年半)数据:
  - 策划配表数据全部加载的话占用内存到99.2M了(luajit是99.2M,对应官方lua是155.7M)，需要优化

Lua优化

- 提取公共结构。在同一个表中共享。
  - 从99.2M优化到70.3M，优化掉了30M（只提取空table｛｝就优化了10多M）
- 结构bean中所有bool类型数压缩成一个int。当然程序接口仍然不变。比如其中一行是｛100, true, false, "abc", true, true｝,压缩后变成｛100, 0xb, "abc"｝，4个bool变成一个int。
  - 内存变为68.4M降了2M，不明显
- 列模式，对int,bool做压缩。当然程序接口仍然不变。如果一个表有5列10000行，那存成lua代码就是 5 行，每行的table里有10000，那么10000个bool，只用存10000/53=189个整数就够了.
  - 自动分析，列模式可减内存时使用列模式，内存占用从68.4M降为54.2M

Configgen的实现

- 4个树结构
  - Define
    - 对应config.xml
  - Data
    - 对应config目录下所有的csv文件，读出来是List<List<String>>
  - Type
    - 从Define中，解析出类型，建立好链接（外键）关系
  - Value
    - 根据Type把Data中的String，读取成带类型的数据，即包括基本类型数据int，string，bool，也包括bean，list，map结构数据。（包括一格复杂结构也要读取出来，隐藏compressAsOne这种细节）

处理流程

- 读取config.xml, csv文件，生成FullDefine，FullType，Data。
- 然后根据Data, csv前2行里的信息来自动完善FullDefine， FullType （补充column，猜测type），之后保存完善后的Define到config.xml，同时让Data内含自己的FullType
- 如有部分提取，根据Full Define和own生成提取后SubType
- 根据SubType和Data生成Value，检验实际数据的约束
- 根据SubType和Value生成各语言的代码和数据文件

杂项

- 客户端lua的热更新
  - 分了好几个.zip，而不是以.luac文件为单位，减少文件个数。
  - 但为了减少热更大小，对.zip，热更系统只更新.zip中改变的.luac。

- Configgen还支持生成c#，但不支持生成c++
  - 求贡献

QA