

# Civilization V (BNW)

## Leader Dialogue Modding

*V5 — Field-tested across nine shipped leaders. Covers the routing system, per-leader Firaxis quirks, the SQLite uniqueness error, wrapper failures, the malformed-tag trap, slot saturation, writing-direction taxonomy, targeted rewrite workflow, and a playbook that scales from one leader to a whole roster.*

This is the reference for overriding existing leader dialogue in Civ 5 BNW. Examples use LEADER\_EXAMPLE as a placeholder — substitute your target leader's actual LeaderType.

**New in V5:** Section 10.5 introduces the writing-direction taxonomy — a class of bugs where routing is correct but the line content is written from the wrong speaker perspective. Section 14 adds Pattern 10 (line feels off but routing is correct) and Pattern 11 (occasional generic on cutscene triggers — diagnostic for the bleed floor). Section 16 documents the targeted-rewrite workflow for iterating on individual pools without re-running the full merge pipeline. Section 19 captures lessons from doing full 124-pool facelift mods, including the legacy versus facelift Tag namespace distinction and the iterate-after-shipping reality. Section 4.4 catalog now lists which leaders have been through full facelift treatment.

### 1. The Big Picture

Civ 5 BNW dialogue is driven by a two-layer system. Layer one is the response routing table (Diplomacy\_Responses), which maps situations to leader-specific text keys. Layer two is the localized text pool (Language\_en\_US), which stores the actual lines.

When the AI needs to say something, the engine asks the routing table: "For this leader, in this situation, what text key pattern should I look up?" It finds all keys matching that wildcard pattern and picks one at random. If no leader-specific routing exists, the engine falls back to a generic key shared across all leaders.

Your mod's job is (a) add or replace routing rows so the engine looks up your custom keys for the target leader, and (b) write text into those keys.

#### Response types vs. text keys

- **Response type:** an enum the engine fires based on the situation. RESPONSE\_FIRST\_GREETING, RESPONSE\_DOW\_LAND, etc.
- **Text key:** a string identifier for a localized text row. TXT\_KEY\_LEADER\_EXAMPLE\_FIRSTGREETING\_1, etc.
- **The routing table** connects them via wildcard patterns ending in %.

## Two Tag namespaces

In V5 the trainer distinguishes between two conventions for the Tag names you'll see and create:

- **Legacy / vanilla namespace:** TXT\_KEY\_LEADER\_EXAMPLE\_X\_N. This is what Firaxis ships. Mods that Update existing keys, or that add a handful of new variants, work in this namespace.
- **Facelift namespace:** Response\_LeaderExample\_RESPONSE\_X\_N (CamelCase leader name in the prefix, full RESPONSE\_ prefix on the category). Used when you are writing a complete new pool for a response type rather than overriding existing vanilla text. The namespace is fresh, so it never collides with vanilla — uniqueness errors do not apply to it.

Both namespaces coexist in a typical full-coverage mod. A leader's override file can have Update statements against TXT\_KEY\_LEADER\_EXAMPLE\_GREETING\_POLITE\_HELLO\_1 (legacy) and Row inserts for Response\_LeaderExample\_RESPONSE\_WORK\_WITH\_US\_1 (facelift) in the same Language\_en\_US block. The Diplomacy\_Responses routing rows decide which pool the engine reads.

## 2. Mod Folder Structure

A leader dialogue mod is two files: the .modinfo descriptor and the .xml file with your routing and text. After F7 build in ModBuddy, the deployed mod lives at:

```
Documents\My Games\Sid Meier's Civilization 5\MODS\  
  [Mod Name] (v 1)\  
    [Mod Name].modinfo  
    [Mod Name].xml
```

### ModBuddy properties for the XML file

- **Import into VFS:** False
- **Action:** OnModActivated → UpdateDatabase
- **In mod properties:** AffectsSavedGames: 1

**Silent-failure trap:** None of these three properties are correct by default. ModBuddy creates new XML files with Import into VFS: True and no Action assigned. You have to explicitly set Import into VFS to False, set the Action to OnModActivated → UpdateDatabase, and set AffectsSavedGames to 1 in the mod's properties panel. All three are silent-failure traps — if any one is wrong, the mod builds successfully and shows as enabled in the mods menu, but no XML actually applies in-game. There is no error message. If your mod appears to do nothing despite a clean build and an enabled status, re-verify all three settings before debugging anything else.

**WARNING:** If Import into VFS is True, the engine treats it as a virtual file rather than a database update, and nothing applies. Easy miss after creating a new file.

### 3. The XML Shape

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <Diplomacy_Responses>
    <!-- routing rows go here -->
  </Diplomacy_Responses>
  <Language_en_US>
    <!-- text Updates and Row inserts go here -->
  </Language_en_US>
</GameData>
```

#### Diplomacy\_Responses row format (legacy / single-pool routing)

```
<Row LeaderType="LEADER_EXAMPLE">
  <ResponseType>RESPONSE_FIRST_GREETING</ResponseType>
  <Response>TXT_KEY_LEADER_EXAMPLE_FIRSTGREETING%</Response>
</Row>
```

#### Diplomacy\_Responses row format (facelift / new-pool routing)

```
<Row LeaderType="LEADER_EXAMPLE">
  <ResponseType>RESPONSE_WORK_WITH_US</ResponseType>
  <Response>Response_LeaderExample_RESPONSE_WORK_WITH_US%</Response>
</Row>
```

The Response value points at whichever pool you intend the engine to read. Use the legacy form when you're Update-ing vanilla keys, the facelift form when you're emitting brand-new pools.

#### Language\_en\_US: Update vs Row

Update changes the text of a key that already exists. Use this for vanilla, G&K, or BNW keys:

```
<Update>
  <Where Tag="TXT_KEY_LEADER_EXAMPLE_FIRSTGREETING_1"/>
  <Set Text="Your replacement text here."/>
</Update>
```

Row inserts a brand new text key. Use only for keys that do not exist in any base or DLC file:

```
<Row Tag="TXT_KEY_LEADER_EXAMPLE_EMBASSY_OFFER_1">
  <Text>Your brand new line here.</Text>
</Row>
```

Facelift pool inserts always use Row (the facelift Tag namespace doesn't exist in vanilla, so uniqueness errors don't apply):

```
<Row Tag="Response_LeaderExample_RESPONSE_WORK_WITH_US_1">
  <Text>Your new facelift line here.</Text>
</Row>
```

**WARNING:** Using Row on a vanilla key that already exists triggers a SQLite uniqueness violation and halts processing of your entire Language\_en\_US block. In-game this looks like "mod did nothing." See Section 5 for the full walkthrough. The uniqueness violation is only possible against the legacy namespace — facelift Tag names that don't exist in vanilla are immune.

## 4. Verifying a Leader Before You Build (Step Zero)

Before generating a template for a new leader, extract the actual counts and prefix details from the vanilla files. This takes 30 seconds and replaces any need to trust numbers from this trainer or any other source. Static count tables go stale or carry typos — running the extraction yourself is authoritative.

### 4.1 Find the vanilla dialogue file

The base path is the Civ 5 install directory:

```
Steam\steamapps\common\Sid Meier's Civilization V\Assets\...
```

Leader files live in one of these subpaths depending on which expansion added the leader:

- **Base game:** Gameplay\XML\NewText\EN\_US\LeaderDialog\Civ5\_Dialog\_[Name].xml
- **G&K:** DLC\Expansion\Gameplay\XML\Text\EN\_US\LeaderDialog\Civ5\_Dialog\_[Name].xml
- **BNW:** DLC\Expansion2\Gameplay\XML\Text\EN\_US\LeaderDialog\Civ5\_Dialog\_[Name].xml

Localization fallback: some leaders' English dialogue files are missing or live in language-specific subfolders (FR\_FR, ES\_ES, DE\_DE) instead of EN\_US. The Tag keys are identical across languages, so any localized file works for extracting counts. Examples: Maria Theresa is in ES\_ES, Isabella in FR\_FR, Boudicca in ES\_ES.

### 4.2 Extract counts (Python one-liner)

Run this from a terminal where the vanilla XML file is accessible. It counts every Tag="..." in the file, grouped by category:

```
python3 -c "
import re, sys
content = open(sys.argv[1]).read()
tags = re.findall(r'Tag=\"([^\"]+)\\"', content)
cats = {}
for t in tags:
    parts = t.rsplit('_', 1)
    cat = parts[0] if (len(parts)==2 and
        (parts[1].isdigit() or parts[1] in ('ANGRY','HAPPY','NEUTRAL')) else t
    cats[cat] = cats.get(cat, 0) + 1
for k in sorted(cats): print(f' {k}: {cats[k]}')
" Civ5_Dialog_LEADERNAME.xml
```

If you don't use Python: open the vanilla file in Notepad++. Use Find → Mark → regex mode, search for Tag="([^\"]+)". The status bar will show the total count, and you can eyeball categories by scrolling.

Slower than the script but works.

### 4.3 Check the routing file for the warning prefix

The vanilla dialogue file gives you the counts. The responses file tells you which prefix Firaxis used for warnings and what BNW-era categories are already routed.

- **Base:** Gameplay\XML\Diplomacy\Civ5Diplomacy\_Responses.xml
- **G&K:** DLC\Expansion\Gameplay\XML\Diplomacy\Civ5Diplomacy\_Responses\_Expansion.xml
- **BNW:**  
DLC\Expansion2\Gameplay\XML\Diplomacy\Civ5Diplomacy\_Responses\_Expansion2.xml

Grep the responses file for your leader's LEADER\_NAME. The Response field on each row shows the exact wildcard pattern the engine routes to. If a warning row uses TXT\_KEY\_NAME\_HOSTILE\_AGGRESSIVE\_MILITARY\_WARNING% (no LEADER\_), that's the warning prefix quirk in action. Match it in your mod's Tag names exactly.

### 4.4 Leader catalog (worked examples)

Nine leaders confirmed working in real mods built across multiple sessions. The "Facelift" column marks leaders that have been through the full 124-pool RESPONSE-type facelift treatment (where every BNW response type has a brand-new Response\_Leader\* pool). The others are running the lighter Update-based approach against the legacy TXT\_KEY namespace. Use these as worked examples for pattern-matching new leaders, but always re-extract counts yourself — even this table can carry typos and is no substitute for running the extraction on the actual vanilla file.

Patterns to notice: most leaders drop LEADER\_ in their warning keys (Catherine, Elizabeth, Maria, Theodora, Boudicca, Dido, Maria I), but Isabella keeps it. Wu Zetian has a LeaderType/key mismatch where the LeaderType has an underscore (LEADER\_WU\_ZETIAN) but the text keys do not (LEADER\_WUZETIAN\_). Always verify per leader.

Leader	LeaderType	Warning Prefix	Notable	Facelift
Catherine	LEADER_CATHERINE	CATHERINE_	Drops LEADER_ in warnings	No
Elizabeth	LEADER_ELIZABETH	ELIZABETH_	ATTACKED gap (1, 3 only)	No
Isabella	LEADER_ISABELLA	LEADER_ISABELLA_	Vanilla in FR_FR; keeps LEADER_	No
Wu Zetian	LEADER_WU_ZETIAN	LEADER_WUZETIAN_	LeaderType/key mismatch	No
Maria Theresa	LEADER_MARIA	MARIA_	Vanilla in ES_ES; TRADE_NEEDMORE 4/3/4	Yes
Theodora	LEADER_THEODORA	THEODORA_	TRADE_NEEDMORE 4/3/4	No
Boudicca	LEADER_BOUDICCA	BOUDICCA_	Vanilla in ES_ES; TRADE_NEEDMORE 3/3/4	No
Dido	LEADER_DIDO	DIDO_	TRADE_NEEDMORE 4/3/4	Yes
Maria I	LEADER_MARIA_I	MARIA_I_	AGREE_SHORT 5 (one more than usual)	Yes

## 5. The SQLite Uniqueness Error — The Silent Killer

The bug that costs the most debugging hours.

### 5.1 The symptom

Mod builds clean. You load a game. Some leaders use your custom dialogue. Others fall back to generic across every category. Build log clean. xml.log clean. Mod is enabled.

### 5.2 What's happening

<Row Tag="..."> does a SQL INSERT. If that Tag already exists (because Firaxis shipped it in BNW even though your reference leader's vanilla file does not have it), the INSERT fails with a uniqueness constraint violation. That single failure halts the entire Language\_en\_US block in that file. The error only appears in Database.log.

**Note on namespaces:** This bug class only applies to the legacy TXT\_KEY namespace. Facelift namespace Tag names (Response\_LeaderName\_RESPONSE\_X\_N) don't collide with vanilla, so Row inserts there never trip the uniqueness error. If your mod is pure facelift, skip ahead — you won't hit this. If your mod mixes Update statements against vanilla TXT\_KEY rows AND facelift Row inserts, the Updates are the ones that need attention.

### 5.3 What to look for in Database.log

```
columns Language, Tag are not unique
While executing - 'insert into Language_en_US('Tag', 'Text') values (?, ?);'
In XMLSerializer while inserting row into table insert into Language_en_US
  with values (TXT_KEY_LEADER_[NAME]_GREETING_POLITE_HELLO_1, ...)
```

### 5.4 The fix

Convert every <Row Tag="..."> in the affected file to Update. Update uses SQL UPDATE, which succeeds whether the row exists or not.

FROM:

```
<Row Tag="TXT_KEY_LEADER_EXAMPLE_GREETING_POLITE_HELLO_1">
  <Text>Your line here</Text>
</Row>
```

TO:

```
<Update>
  <Where Tag="TXT_KEY_LEADER_EXAMPLE_GREETING_POLITE_HELLO_1" />
  <Set Text="Your line here" />
</Update>
```

## 5.5 Notepad++ regex for bulk conversion

Find & Replace with Regular expression mode and '. matches newline' both enabled:

Find:

```
<Row Tag="([ ^" ]+)">\s*<Text>(.*?)</Text>\s*</Row>
```

Replace with:

```
<Update><Where Tag="$1"/><Set Text="$2"/></Update>
```

After replacing: do a non-regex search for " and scan the hits — any double-quote inside dialogue text needs to become &quot;; otherwise it closes the Set Text="..." attribute early and breaks the file.

## 5.6 Row vs Update — the decision tree

- **Use Update when:** the Tag exists in vanilla, G&K, or BNW. Safe default for legacy keys — cannot cause uniqueness errors. If the key does not exist, Update silently no-ops (which is bad but not fatal).
- **Use Row when:** (a) you are certain the Tag does not exist anywhere — typically adding variants beyond vanilla counts (like ATTACKED\_4 when vanilla has 3); or (b) you are inserting into the facelift namespace (Response\_LeaderName\_RESPONSE\_X\_N), which is fresh by design.

BNW-era categories (POLITE\_HELLO, NEUTRAL\_HELLO, HOSTILE\_HELLO, EMBASSY\_\*, TRADE\_CANT\_MATCH, REQUEST, LUXURY\_TRADE, OPEN\_BORDERS\_EXCHANGE, MILITARY\_WARNING, EXPANSION\_WARNING) — Firaxis quietly added these for some leaders but not others. Default to Update for these in the legacy namespace. Update no-ops safely if the key does not exist; Row halts the file if the key does.

## 5.7 The malformed-tag trap (WINWAR\_2 bug)

When an AI fills a blank template, it sometimes splits the Update format wrong, producing this pattern (which is invalid XML):

```
<Update><Where Tag="TXT_KEY_LEADER_EXAMPLE_WINWAR_2"/><Set Text="">
Your replacement line text here.</Text>
```

Three things wrong: Set Text="" is empty, the dialogue is outside the attribute, and it closes with </Text> (Row syntax) instead of /></Update>. The parser sees an open Set tag and wanders looking for a close, breaking at the next tag.

The fix: combine into one line with the text inside the attribute:

```
<Update><Where Tag="TXT_KEY_LEADER_EXAMPLE_WINWAR_2"/>
<Set Text="Your replacement line text here."/></Update>
```

To find every instance in a file, use Notepad++ Find with regular text (not regex):

```
<Set Text="">
```

Every hit is the same bug. Each needs the same one-line fix.

Preventive prompt language when delegating template fill to another AI: include the explicit instruction that for Update entries, dialogue text **MUST** go inside the Set Text="..." attribute on a single line ending with "></Update>", never split across lines.

## 6. XML Structure — Wrapper and Parse Errors

### 6.1 The required wrapper

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <Diplomacy_Responses>
    <!-- routing rows -->
  </Diplomacy_Responses>
  <Language_en_US>
    <!-- Updates and Rows -->
  </Language_en_US>
</GameData>
```

### 6.2 'Junk after document element' error

Two elements at the top level without a parent containing them — usually Diplomacy\_Responses and Language\_en\_US sitting side-by-side without the GameData wrapper. Fix by adding the wrapper.

### 6.3 Character escaping in text content

Character	Escape	Why
&	&amp;	Otherwise treated as start of entity reference
<	&lt;	Otherwise treated as start of a tag
>	&gt;	Pair with &lt; (safe everywhere)
" (inside attribute)	&quot;	Required inside Set Text="..." attributes
' apostrophe	(rephrase)	Civ 5 font renders curly apostrophes as garbage; use straight ASCII or rephrase
— em dash	-- (double-hyphen)	Em dashes render poorly in Civ 5 font
“ ” curly quotes	plain "	Curly/smart quotes render as garbage
... ellipsis	...	Unicode ellipsis renders as garbage
ö / ä / ü etc.	o / a / u	Civ 5 font renders non-ASCII inconsistently — sometimes as box or ? glyph



## 7. Per-Leader Firaxis Quirks

Civ 5's dialogue tables were built across base game, G&K, and BNW over many years. The result is per-leader inconsistencies that cause silent fallback bugs.

### 7.1 The warning prefix quirk

Most leader text keys follow `TXT_KEY_LEADER_[NAME]_[CATEGORY]_N`. But `MILITARY_WARNING` and `EXPANSION_WARNING` for many leaders drop the `LEADER_` entirely: `TXT_KEY_[NAME]_HOSTILE_AGGRESSIVE_MILITARY_WARNING_N`. From the catalog: Catherine, Elizabeth, Maria Theresa, Theodora, Boudicca, Dido, and Maria I all drop the `LEADER_` prefix on warnings. Isabella keeps it. Always check the routing file.

Failure if you get this wrong: warnings silently fall back to generic. No build error.

### 7.2 LeaderType vs key prefix mismatch (Wu Zetian)

Most leaders have a `LeaderType` matching their text key prefix exactly. Wu Zetian is the exception:

- **LeaderType in routing:** `LEADER_WU_ZETIAN` (with underscore)
- **Text key prefix:** `LEADER_WUZETIAN_` (no underscore)

If you 'fix' one to match the other, routing breaks and Wu Zetian falls back to generic. Preserve the mismatch.

### 7.3 Variant numbering gaps

Some leaders have non-contiguous variant numbering. Elizabeth's `ATTACKED` has only `_1` and `_3` — no `_2`. If you write an Update for `ATTACKED_2` on Elizabeth, SQLite silently no-ops because there is no row to update. Your line never plays. Use Row to insert it as a new key.

### 7.4 Uneven mood splits

`TRADE_NEEDMORE` comes in three moods (`ANGRY`, `HAPPY`, `NEUTRAL`). Counts per mood are not always balanced. Examples: Maria Theresa, Theodora, Dido, and Maria I had 4/3/4 splits. Boudicca had 3/3/4. Count each mood separately when extracting.

## 8. DLC Leaders and the Foreign-Language Workaround

Some DLC leaders' English dialogue files are missing or hard to find. Their localized versions (FR\_FR, ES\_ES, DE\_DE, etc.) have the same Tag keys but in another language. Extract counts from the localized file, ignore the foreign text, and write your own.

Examples: Isabella was only in FR\_FR. Maria Theresa was in ES\_ES rather than EN\_US. Boudicca was in ES\_ES. Tag names are ASCII and identical across all languages, so this is purely an inconvenience, not a blocker.

## 9. The Delete Trap (User Mods)

Firaxis files sometimes use `<Delete LeaderType="..." />` followed by Row inserts to replace routing. This works for Firaxis because their files load differently at engine startup. In user mods loaded via `OnModActivated` → `UpdateDatabase`, Delete wipes out vanilla routing, but your Row inserts only cover the ResponseTypes you explicitly list. Every response type you forgot now falls back to generic.

### Targeted-Delete corollary

A more surgical-looking variant exists in some documentation:

```
<Delete LeaderType="LEADER_EXAMPLE" ResponseType="RESPONSE_EXPANSION_WARNING" />
```

This appears nowhere in vanilla Firaxis files. In testing, it either silently no-ops or has unpredictable behavior depending on the engine version. Treat it the same as the blanket form: avoid in user mods.

### Rule for user mods

Avoid Delete and Replace entirely. Add plain Row inserts to `Diplomacy_Responses`, and use Update statements in `Language_en_US` for existing keys. If vanilla has a competing routing row that you cannot remove, accept that the engine will sometimes pick the vanilla pool — this is part of the bleed floor described in Section 15.

## 10. Response Type Taxonomy

The BNW ResponseTypes worth knowing, grouped by category. Use as a checklist for full-coverage mods. The trigger-direction annotations in Section 10.5 below are critical for writing lines that match the situation the engine actually fires them in.

## 10.1 Greetings

- RESPONSE\_FIRST\_GREETING — first meeting, plays once
- RESPONSE\_GREETING\_POLITE\_HELLO / \_NEUTRAL\_HELLO / \_HOSTILE\_HELLO — mood-based
- RESPONSE\_GREETING\_REPEAT / \_HOSTILE\_REPEAT — already talked recently
- RESPONSE\_REPEAT\_TOO\_MUCH / \_GREETING\_REPEAT\_TOO\_MUCH — clicked too many times

## 10.2 BNW situational greetings

- RESPONSE\_GREETING\_RESEARCH\_AGREEMENT, \_WORKING\_WITH, \_WORKING\_AGAINST, \_COOP\_WAR
- RESPONSE\_GREETING\_HUMAN\_AT\_WAR, \_HOSTILE\_HUMAN\_AT\_WAR
- RESPONSE\_GREETING\_AGGRESSIVE\_MILITARY / \_EXPANSION / \_PLOT\_BUYING and \_HOSTILE\_ variants
- RESPONSE\_GREETING\_FRIENDLY\_STRONG\_MILITARY / \_STRONG\_ECONOMY
- RESPONSE\_GREETING\_HOSTILE\_HUMAN\_FEW\_CITIES / \_SMALL\_ARMY / \_IS\_WARMONGER
- RESPONSE\_GREETING\_DENOUNCED\_BY\_AI / \_AI
- RESPONSE\_GREETING\_OUR\_DOF\_WITH\_ENEMY\_OF\_AI / \_FRIEND\_OF\_AI
- RESPONSE\_GREETING\_DENOUNCED\_FRIEND\_OF\_AI / \_ENEMY\_OF\_AI

## 10.3 Warnings, Trade, Tribute, War, Peace, Short Responses

- **Warnings:** RESPONSE\_HOSTILE\_AGGRESSIVE\_MILITARY\_WARNING / \_AGGRESSIVE\_MILITARY\_WARNING, \_EXPANSION\_SERIOUS\_WARNING / \_EXPANSION\_WARNING
- **Trade:** \_REQUEST, \_LUXURY\_TRADE, \_OPEN\_BORDERS\_EXCHANGE, \_TRADE\_AI\_MAKES\_OFFER, \_DEMAND, \_EMBASSY\_OFFER, \_EMBASSY\_EXCHANGE, \_TRADE\_ACCEPT\_\* / \_TRADE\_REJECT\_\*, \_TRADE\_DEAL\_UNCHANGED, \_NO\_DEAL\_POSSIBLE, \_TRADE\_CANT\_MATCH\_OFFER
- **Tribute:** \_HUMAN\_DEMAND\_YES, \_HUMAN\_DEMAND\_REFUSE\_WEAK / \_HOSTILE / \_TOO\_MUCH / \_TOO\_SOON
- **War:** \_ATTACKED\_\* (9 variants), \_DOW\_GENERIC / \_LAND / \_WORLD\_CONQUEST / \_OPPORTUNITY / \_DESPERATE, \_DOW\_BETRAYAL / \_WEAK\_BETRAYAL / \_REGRET, \_WAR\_DEMAND\_REFUSED
- **Peace / war state:** \_PEACE\_OFFER, \_PEACE\_MADE\_BY\_HUMAN\_GRACIOUS, \_NO\_PEACE / \_TOO\_SOON\_NO\_PEACE, \_GREETING\_WILL\_ACCEPT\_SURRENDER / \_AT\_WAR\_HOSTILE / \_WANTS\_PEACE / \_DESTRUCTION\_LOOMS
- **Short responses:** \_LETS\_HEAR\_IT / \_DOT\_DOT\_DOT, \_SO\_BE\_IT, \_WORK\_WITH\_US\_YES/NO, \_WORK\_AGAINST\_SOMEONE\_YES/NO, \_DONT\_SETTLE\_YES/NO, \_COOP\_WAR\_YES/NO, \_REPEAT\_NO, \_HUMAN\_RESPONSE\_PLEASSED / \_DISAPPOINTED / \_THANKFUL
- **Defeated:** \_DEFEATED — when you conquer the leader

## 10.4 Spy / intrigue category (call out)

Spy response types are the most commonly misread. Worth specific attention:

- **RESPONSE\_CAUGHT\_YOUR\_SPY** — SHE caught YOUR spy. She's the captor, addressing you about the catch.
- **RESPONSE\_KILLED\_YOUR\_SPY** — SHE killed YOUR spy (typically during your failed steal attempt). She's the actor.
- **RESPONSE\_KILLED\_MY\_SPY** — HER spy was killed in YOUR territory. She's confronting you as the responsible party. The naming is from her perspective: "my spy was killed," with YOU being the implicit killer.
- **RESPONSE\_CONFRONT\_YOU\_KILLED\_MY\_SPY** — YOU came to her diplomatic screen to address the spy incident. She's acknowledging your visit. The trigger is the confrontation, not literal death; her spy may have been caught/exposed rather than killed.
- **RESPONSE\_HUMAN\_KILLED\_MY\_SPY\_FORGIVEN / \_UNFORGIVEN** — Her reply to your forgive / don't-forgive choice in the confrontation flow.
- **RESPONSE\_STOP\_SPYING\_YES / \_NO** — Her reply when YOU ask her to stop spying.
- **RESPONSE\_SHARE\_INTRIGUE\_\*** — SHE proactively comes to YOU to share intelligence about a third party.

CONFRONT\_YOU\_KILLED\_MY\_SPY and KILLED\_MY\_SPY are particularly easy to get wrong because the names suggest the same scenario. They are distinct triggers and need distinct line content.

## 10.5 Trigger direction taxonomy

Response type names can be misleading about who is speaking and who is being addressed. Getting this wrong produces lines that read backwards in-game — routing is fine, content is for the wrong direction. The four direction categories:

Category	Pattern	What lines should sound like
AI-initiates	SHE proactively comes to YOU with a proposal, accusation, warning, or news.	Her voice asking/proposing/warning/informing. She is the speaker; you are the audience. Example trigger: RESPONSE_WORK_WITH_US (her DoF pitch to you).
AI-reply	SHE replies to something YOU just did or said.	Her voice reacting — accepting, refusing, expressing approval/disapproval. Short or measured tone. Example: RESPONSE_WORK_WITH_US_YES (her agreeing after you proposed DoF).
AI-confront	SHE calls YOU out for an action you took, OR you come to her about something she did.	Direct address, naming the act, demanding/explaining. Example: RESPONSE_KILLED_MY_SPY (she confronts you about her spy you killed), RESPONSE_CONFRONT_YOU_KILLED_MY_SPY (you come to her, she acknowledges).
AI-acts	SHE describes an action she has taken or is taking.	Her voice as the actor — bragging, explaining, justifying. Example: RESPONSE_KILLED_YOUR_SPY (she narrates killing your spy).

## Common direction-confusing names

Response type	Sounds like	Actually fires when
WORK_WITH_US	Her accepting your DoF proposal	SHE comes to YOU pitching a DoF. Lines should be written as her proposing, not accepting.
WORK_WITH_US_YES	Same as above	SHE accepts YOUR DoF proposal. Typically routed to her short-agree pool (one-line response, not a speech).
KILLED_MY_SPY	She mourns a dead spy	SHE confronts YOU about killing her spy. Lines should be addressed to you as the killer, not narrate her loss.
CONFRONT_YOU_KILLED_MY_SPY	She accuses you of killing her spy	YOU came to HER about a spy incident. Lines should acknowledge your visit, not lament a death.
HUMAN_EXPANSION_WARNING_BAD	Your hostile reply to her warning	HER warning to you, delivered in a bad mood. The GOOD/BAD suffix describes HER mood, not your response.
DOF_AI_DENOUNCE_REQUEST	She asks you to denounce someone	Correct as named — SHE proposes the denounce. Use as a worked example of a name that does mean what it says.
RENEW_DEAL	She accepts a deal renewal	SHE comes to you proposing to renew an expiring deal. Lines should be her pitching the renewal.
OPEN_BORDERS_OFFER	She accepts your open-borders offer	SHE offers open borders to you. Lines should be her making the offer.

## Diagnostic ladder for suspected direction issues

- 1. The line that fires sounds backwards or addressed to the wrong audience.
- 2. Identify which response type is firing (use the line content to map back to your pool).
- 3. Look up the trigger direction in 10.5 (or the routing file).
- 4. If direction mismatches your written content: bug is in writing, not routing. Rewrite the pool from the correct speaker perspective using the targeted-rewrite workflow in Section 16.
- 5. If direction matches but the line still feels off: re-test multiple times. May be the bleed floor from Section 15.

## 11. The Labeled-List Workflow (Writing Dialogue at Scale)

After 9 leaders, the fastest workflow for writing dialogue turned out to be a three-step pipeline that separates structural work from creative writing. Trying to write lines directly into XML is slow and error-prone; trying to write lines into an XML template introduces formatting bugs.

### 11.1 Step 1: build the blank XML template

Extract vanilla counts (Section 4), confirm warning prefix and LeaderType from routing, then build a blank template with empty Set Text="" attributes for every slot. This step is mechanical and a small amount of automation handles it well.

### 11.2 Step 2: write dialogue in a plain labeled list

Generate a flat text list with one label per slot:

```
FIRSTGREETING_1:
FIRSTGREETING_2:
FIRSTGREETING_3:
GREETING_1:
GREETING_2:
... etc
```

Fill the lines in plain text, no XML syntax to worry about. This is where you spend creative energy — voice, mood variation across the three mood-based categories, tonal range from polite to hostile to surrender. Apply the apostrophe-expansion and em-dash rules at this stage so you do not have to clean it up later.

### 11.3 Step 3: have an AI paste lines into the template

Give the AI the blank template plus the filled labeled list with a prompt covering: tag mapping, apostrophe rules, escape rules, the warning-prefix quirk, the TRADE\_NEEDMORE split, wrapper preservation, validation, and the malformed-tag warning. A good prompt template covers all of these in one block.

Why this works better than writing into XML directly: separation of concerns. The creative work is in plain text. The structural work is the template. The integration is mechanical. Errors that happen tend to localize to one step.

### 11.4 Direction-aware label format (recommended for facelift mods)

When the goal is a full facelift (every BNW response type) rather than overrides on legacy keys, add a one-line direction note next to each label so the dialogue gets written with the right speaker perspective the first time. Example:

```
WORK_WITH_US (SHE pitches DoF to YOU – her proposal voice):  
  Line 1  
  Line 2  
  Line 3  
  Line 4  
  
CONFRONT_YOU_KILLED_MY_SPY (YOU came to HER about a spy incident – she acknowledges):  
  Line 1  
  Line 2  
  Line 3  
  Line 4  
  
KILLED_MY_SPY (HER spy died in YOUR territory – she confronts you as the responsible party):  
  Line 1  
  Line 2  
  Line 3  
  Line 4
```

This costs nothing in the writing phase and saves rewriting the same pools after in-game testing. See Section 19 for why iterating after shipping is the norm rather than the exception.

## 12. Formatting Traps When Delegating to Another AI

### 12.1 Wrapper tags outside the code block

Some AIs return the XML by putting the `<?xml ... ?>` and `<GameData>` wrapper tags as plain text above the code block, then the inner content inside the code block, then the closing `</GameData>` below. When you copy just the code block, you get a file missing its wrapper.

Fix: add to your prompt the explicit instruction that the ENTIRE file, including `<?xml>` declaration and both `<GameData>` tags, must be inside ONE code block. If it happens anyway, the easy workaround is to manually add the two wrapper lines in Notepad++ — 10 seconds of typing.

### 12.2 Syntax highlighting is not validation

If the file content looks plain (uncolored) in the chat code block while other leaders' files were color-coded, that means the chat renderer did not tag the code block as XML — not that the file is broken. Color coding depends on the AI explicitly using ````xml` as the fence opener, which varies by AI and prompt phrasing.

The real validation test is ModBuddy F7. If the file builds, the XML is valid. If it errors, the error message includes a line number you can fix from. Save the file, drop it in, press F7, see what happens. That is more reliable than judging by chat rendering.

## 13. Build, Test, Debug Workflow

### 13.1 ModBuddy properties

- Import into VFS = False
- Action (OnModActivated) = UpdateDatabase
- AffectsSavedGames = 1 (in mod properties)

### 13.2 The cache deletion dance

After every rebuild, delete:

```
Documents\My Games\Sid Meier's Civilization 5\cache\Civ5ModsDatabase.db
```

If you skip this, changes may not take effect even though the .modinfo says it's the latest version.

### 13.3 Logs to check after a failed test

```
# Database.log -- SQL errors, uniqueness violations
Documents\My Games\Sid Meier's Civilization 5\Logs\Database.log

# xml.log -- XML parse errors
Documents\My Games\Sid Meier's Civilization 5\Logs\xml.log
```

Enable logging if these are empty — add to config.ini:

```
LoggingEnabled = 1
MessageLog = 1
ValidateGameDatabase = 1
```

## 14. Debugging Cookbook

### Pattern 1: Leader falls back to generic ALL categories

Most likely: SQLite uniqueness error (Section 5). Check Database.log for 'columns Language, Tag are not unique'. Convert Rows to Updates with the regex. Also possible: LeaderType mismatch (Wu Zetian quirk), or wildcard prefix mismatch between routing and language keys.

### Pattern 2: Specific categories fall back (warnings)

Cause: warning prefix mismatch (Section 7.1). Either the routing uses LEADER\_NAME\_ but your Row Tag uses NAME\_ (or vice versa). Both halves must match what's in the vanilla routing file.

### Pattern 3: One variant never appears

Cause: Update used for a Tag that doesn't exist (numbering gap like Elizabeth ATTACKED\_2). Use Row instead.



## Pattern 4: 'Tag was not closed' parse error

Cause: Malformed Set tag (Section 5.7 — WINWAR\_2 bug). Search for `<Set Text="">` in the file. Every hit is the same bug, same one-line fix.

## Pattern 5: 'Junk after document element' parse error

Cause: Missing or duplicated GameData wrapper. Or the AI that generated the file put the wrapper outside the code block and you missed copying it (Section 12.1).

## Pattern 6: Build succeeds but mod won't enable

Most likely: stale cache. Delete Civ5ModsDatabase.db. Also check `AffectsSavedGames = 1`.

## Pattern 7: Random character glitches in dialogue

Cause: unescaped apostrophes (font glitch) or em dashes / smart quotes / ellipsis / non-ASCII letters. Expand contractions, rephrase possessives, use ASCII (Section 6.3).

## Pattern 8: Generic dialogue bleeds through occasionally

Some response types route through generic keys at the DLL level. Add more situational greeting routes (Section 10). The ~10% generic bleed is the floor for XML-only mods — to reduce perceived bleed, saturate your routed slots past vanilla counts (Section 15.1 and 15.2).

## Pattern 9: Previously-working mod silently stops working

A leader's mod was working in earlier playthroughs and now produces no dialogue (silent leader, vanilla fallback across all categories, or mod fails to apply). Database.log shows nothing relevant. No changes were made to the mod itself.

Cause: ModBuddy build IDs and cache state can drift out of sync when other mods in the project are rebuilt, even on untouched leaders. The cache may be pointing at a stale build version that no longer matches what the engine is looking for.

Fix: open the affected leader's project in ModBuddy, rebuild (F7) even with no changes, delete Civ5ModsDatabase.db (Section 13.2), then test again. Cheap fix, common cause, worth attempting before deeper debugging when nothing else explains the failure.

## Pattern 10: In-game line feels off but routing is correct

Symptom: A line fires that's clearly from your custom pool — not vanilla, not generic — but it sounds backwards or addressed to the wrong audience. Common examples: a DoF pitch that reads like an acceptance speech, a spy-incident line that mourns a death when no one died, a warning reply written as a warning instead of a reply.

Cause: Writing direction mismatch (Section 10.5). The response type name misled the writer about who is speaking or what the trigger is. Routing is fine, content is for the wrong direction.

Fix: Look up the trigger direction in the Section 10.5 table. Confirm which response type is actually firing by matching the line content to your pool. Rewrite the 4 lines from the correct speaker perspective using the targeted-rewrite workflow in Section 16. No XML structural changes needed.

## Pattern 11: Generic on cutscene trigger, custom on follow-up

Symptom: She comes to your throne room about something (expansion warning, military buildup, spy incident) and delivers a generic / vanilla-sounding line. You pick a response option, and her reply IS in your custom voice.

Cause: Bleed floor on the cutscene trigger (Section 15 main text). Some response types — particularly the proactive-visit / warning categories — partially bypass the routing table at the DLL level. Your routing handles the follow-up response type correctly, but the initial cutscene line has its own engine-level path.

Diagnostic: this pattern (generic first, custom second in the same conversation) is the signature of bleed, not a routing bug. If you re-trigger the same cutscene multiple times across a playthrough, you'll get a mix of custom and generic on the initial line — that's the ~10% floor. If it's generic every single time, it may be a hard DLL-direct lookup for that category and Section 15.2 saturation on adjacent slots is the only XML-side mitigation.

## 15. The Generic Fallback Floor

Even with full routing coverage, some lines still come from generic text keys the BNW DLL looks up directly without routing. Examples: 'Anything else?', 'What would you like to discuss?', one-word fillers in certain contexts.

These use keys like `TXT_KEY_DIPLOMACY_ANYTHING_ELSE`. Adding leader-specific versions does not intercept the lookup — the DLL reads the generic keys directly.

Two practical options:

- **Accept it.** ~10% generic bleed is the floor. Best for most use cases.
- **Saturate the slots you control (Section 15.1).** Reduces perceived bleed by making routed lines hit much more often than fallback lines. The most effective option for XML-only mods.

A third option exists in theory — overriding the generic keys directly — but it changes those lines for every leader in the game, including ones the mod doesn't otherwise touch. Reasonable for whole-roster mods (where you control all leaders' tone), but not recommended for single-leader projects.

DLL modding could remove the floor entirely but is a massive complexity jump beyond XML and outside the scope of this trainer.

### 15.1 Why Saturation Reduces Perceived Bleed

The 10% generic-bleed floor describes lines the DLL pulls outside the routing system. But the perceived bleed rate — how often a player actually hears a generic line — depends on how often the engine has to fall back at all. The more custom lines per routed slot, the less the generic pool gets used, and the more polished the mod feels in play.

The mechanic: when the engine routes a response through your wildcard pattern (e.g., `TXT_KEY_LEADER_EXAMPLE_GREETING_POLITE_HELLO%`), it picks one of the matching keys at random. If vanilla shipped 4 lines for that slot and you only updated those 4, your custom pool is 4 lines deep. Add Row inserts for `_5` through `_12`, and your custom pool is now 12 lines deep. Same wildcard, same routing — the engine just has more to choose from, and the proportion of plays that come from your custom content rises accordingly.

What this does not fix: DLL-level generic lookups (Section 15 main text). Those still bleed regardless of how saturated your routed slots are. The point of saturation is to make those generic moments feel like rare exceptions rather than the dominant tone.

## 15.2 The Saturation Workflow

Slot saturation works as a separate expansion pass after a leader's baseline mod is built and tested. It is not a first-pass activity — get the baseline shipping cleanly first (Sections 4 through 13), then layer saturation on top.

### Step 1: Audit your own mod's slot counts.

Run the Section 4.2 extraction script against your own mod file (not the vanilla file). The output gives you per-slot counts of what your mod currently provides. Compare to vanilla counts from Section 4.2 run on the vanilla file. Slots at parity with vanilla are candidates for expansion.

### Step 2: Identify high-impact slots.

Some slots have outsized impact on perceived quality:

- Greeting slots (POLITE\_HELLO, NEUTRAL\_HELLO, HOSTILE\_HELLO) — players see these dozens of times per game.
- Trade and tribute mood slots (TRADE\_NEEDMORE\_\*, TRADE\_YES\_\*, TRIBUTE\_\*) — frequent in active diplomatic games.
- WARBLUFF — fires on every refused war demand.
- AGREE\_SHORT, DISAGREE\_SHORT, LETSHEARIT — short responses that fire many times per conversation.

Low-impact slots not worth expanding:

- FIRSTGREETING — plays once per game per leader.
- WINWAR — plays once when the leader wins a war against the player.
- DEFEATED — plays once when conquered.

Single-fire slots benefit from one or two strong lines, not deep pools.

### Step 3: Add Row inserts past the vanilla count.

For each high-impact slot you decide to expand, write new lines as Row inserts using numbers beyond what vanilla shipped. If vanilla has \_1 through \_4, your additions start at \_5. These keys don't exist anywhere, so they won't trigger the SQLite uniqueness error from Section 5 — Row is correct here, not Update.

The wildcard match in the routing table picks them up automatically. No changes to Diplomacy\_Responses are needed when you add new variants of an already-routed slot.

#### Step 4: Cascade routing for repeat states.

RESPONSE\_REPEAT\_TOO\_MUCH and RESPONSE\_GREETING\_REPEAT\_TOO\_MUCH fire when the player clicks the same conversation option too many times. By default these can fall back to generic pools. Route them into your saturated hello pool instead:

```
<Row LeaderType="LEADER_EXAMPLE">
  <ResponseType>RESPONSE_REPEAT_TOO_MUCH</ResponseType>
  <Response>TXT_KEY_LEADER_EXAMPLE_GREETING_NEUTRAL_HELLO%</Response>
</Row>
```

This redirects one of the most common bleed paths back into your custom content. Useful even when the rest of the leader is at vanilla counts.

#### Step 5: Test and re-audit.

After the expansion pass, rebuild, clear the cache (Section 13.2), and play through a few diplomatic conversations. If a category still feels repetitive, run the extraction script on your mod again and check whether that slot needs another pass.

A practical target for a saturated leader is roughly 1.5x to 3x vanilla coverage on high-impact slots. Below 1.5x the player still hears repetition; above 3x diminishing returns set in quickly.

What saturation cannot mitigate:

- The DLL-level generic keys from Section 15 — those still bleed at their fixed rate regardless of slot depth.
- Slots that don't route through wildcards (single-key responses bound to fixed tags).
- Writing-direction issues (Section 10.5) — adding more lines in the wrong direction just gives the engine more wrong-direction lines to pick from. Direction is a writing problem, not a saturation problem.

## 16. Targeted Rewrites Without Full Re-merge

When in-game testing reveals a single pool needs rewriting (typically 4 lines for a single response type), there is no need to re-run the full template-fill / merge pipeline from Section 11. The change is mechanical: replace the `<Text>...</Text>` contents of the 4 affected Row entries in the deployed XML file, keeping everything else untouched.

### 16.1 The workflow

- Identify the response type whose pool needs rewriting (e.g., RESPONSE\_WORK\_WITH\_US, RESPONSE\_CONFRONT\_YOU\_KILLED\_MY\_SPY).
- Write 4 new lines in plain text using the labeled-list format from Section 11.2, with the direction-aware label from Section 11.4 to keep speaker perspective correct.
- Apply the character normalization rules (Section 6.3) at the writing stage — straight ASCII apostrophes, double-hyphen for dashes, plain quotes, no ellipsis Unicode character, ASCII-only letters.
- For each of the 4 lines, find the matching Row in the deployed XML and replace just the inner `<Text>...</Text>` content.

- Validate the resulting XML (xmllint --noout or just open in Notepad++ and check the syntax highlighting holds).
- Drop the new file in, clear the cache (Section 13.2), test in-game.

## 16.2 Why this works

The mod's XML file is essentially a flat key-value store: each `<Row Tag="...">` is a self-contained entry. Editing one Row's text content has no effect on any other Row, on any routing entries, or on any of the other 100+ Updates in the file. The targeted swap is the minimum possible change.

Compared to re-running the full pipeline: no template regeneration, no merge audit, no risk of accidentally clobbering existing rows during a fresh insertion. The blast radius is exactly the 4 lines being changed.

## 16.3 What to avoid

- Do not change the Tag attribute when rewriting — only the `<Text>` content. The Tag connects to the routing wildcard, and changing it orphans the line.
- Do not change the surrounding XML structure (the `<Row Tag="...">` wrapper). Apply edits only to the text between `<Text>` and `</Text>`.
- Do not re-run the full template fill to get "new versions" of unaffected pools. The unaffected pools are already correct; regenerating them risks introducing fresh malformed-tag bugs (Section 5.7) for no gain.
- Do not skip character normalization on the new lines just because the rest of the file is already clean. New content can introduce smart quotes from any source it was drafted in.

## 16.4 Iteration cadence

Expect to do several targeted rewrites per leader as in-game testing reveals direction mismatches that were not obvious from the labeled list. The first pass catches the structurally wrong lines (clearly the wrong category being addressed); subsequent passes catch the subtle direction issues (pool fires correctly for the trigger, but the lines were written from the wrong speaker perspective).

A typical post-ship iteration log for one leader might look like: rewrite `WORK_WITH_US` after the DoF read backwards; rewrite `CONFRONT_YOU_KILLED_MY_SPY` after a successful enemy steal triggered confusion-mourning lines; rewrite `RENEW_DEAL` after she pitched a renewal but the lines read as an acceptance. Each rewrite is 4 lines, takes 10-15 minutes, and ships as a point release.

## 17. Working Minimal Template

First greeting, three hello moods, defeated. Substitute LEADER\_EXAMPLE.

```
<?xml version="1.0" encoding="utf-8"?>
<GameData>
  <Diplomacy_Responses>
    <Row LeaderType="LEADER_EXAMPLE">
      <ResponseType>RESPONSE_FIRST_GREETING</ResponseType>
      <Response>TXT_KEY_LEADER_EXAMPLE_FIRSTGREETING%</Response>
    </Row>
    <Row LeaderType="LEADER_EXAMPLE">
      <ResponseType>RESPONSE_GREETING_POLITE_HELLO</ResponseType>
      <Response>TXT_KEY_LEADER_EXAMPLE_GREETING_POLITE_HELLO%</Response>
    </Row>
    <Row LeaderType="LEADER_EXAMPLE">
      <ResponseType>RESPONSE_GREETING_NEUTRAL_HELLO</ResponseType>
      <Response>TXT_KEY_LEADER_EXAMPLE_GREETING_NEUTRAL_HELLO%</Response>
    </Row>
    <Row LeaderType="LEADER_EXAMPLE">
      <ResponseType>RESPONSE_GREETING_HOSTILE_HELLO</ResponseType>
      <Response>TXT_KEY_LEADER_EXAMPLE_GREETING_HOSTILE_HELLO%</Response>
    </Row>
    <Row LeaderType="LEADER_EXAMPLE">
      <ResponseType>RESPONSE_DEFEATED</ResponseType>
      <Response>TXT_KEY_LEADER_EXAMPLE_DEFEATED%</Response>
    </Row>
  </Diplomacy_Responses>
  <Language_en_US>
    <Update><Where Tag="TXT_KEY_LEADER_EXAMPLE_FIRSTGREETING_1"/>
      <Set Text="Your first greeting line here."/></Update>
    <Row Tag="TXT_KEY_LEADER_EXAMPLE_EMBASSY_OFFER_1">
      <Text>Your invented embassy offer line here.</Text>
    </Row>
  </Language_en_US>
</GameData>
```

## 18. The Replication Speed Gain

The first leader you mod takes the longest. The second takes a fraction of the time. By the ninth, the per-leader build is mostly mechanical.

The structural XML stays nearly identical across leaders. What changes is the LeaderType, the key prefix, the warning prefix (with the quirk), the per-leader vanilla counts (extract from the file), and the dialogue text itself. The first leader teaches you the engine quirks. After that the per-leader build is a find-and-replace plus the labeled-list writing workflow.

Approximate per-leader time after the playbook was established: 30-60 minutes of structural and template work for an Update-based override against the legacy namespace; longer for a full facelift (Section 19). In both cases, most of the project's clock time is dialogue writing, not modding.

## 19. Full-Facelift Mods (lessons)

A full facelift means writing brand-new pools for every BNW response type (~124 categories, ~496 lines at 4-per-pool) and routing them via the facelift Tag namespace (Response\_LeaderName\_RESPONSE\_X\_N). This is a different scale of project from the Update-based override and runs into a different set of failure modes worth flagging.

### 19.1 Why facelift instead of Update

Update-based overrides only touch lines that already exist as TXT\_KEY rows in vanilla. For leaders Firaxis under-shipped on, that leaves many response types (the BNW-era situational greetings, certain warnings, the spy/intrigue category) with no leader-specific text at all — they fall through to GENERIC routing in the engine. Adding leader-specific Row inserts in the legacy namespace works but runs the SQLite uniqueness gauntlet (Section 5).

The facelift approach sidesteps the uniqueness problem entirely: brand-new Tag namespace means no collisions are possible. The trade-off is that you must also add routing rows for every facelift pool — without them, your lines exist but are never fetched.

### 19.2 The pool-with-no-routing failure

The most common facelift bug: you generate 124 pools (496 language Rows) and ship without generating the 124 corresponding routing Rows. The mod loads cleanly. Database.log shows nothing. In-game every response type falls back to vanilla generic because the engine never knows your custom pools exist.

Diagnostic: search the deployed XML for the count of Response\_LeaderName\_RESPONSE\_ Tag entries (should be  $4 \times N$  for an N-pool facelift) and the count of routing rows with Response patterns matching Response\_LeaderName\_RESPONSE\_%. The two counts must be: language Tag count =  $4 \times$  routing row count (if every pool has exactly 4 lines). Mismatch = orphan pools or dangling routing.

Prevention: build the pipeline to emit both halves from the same source draft. If you only have a script that emits language Rows, you have a script that ships broken mods.

### 19.3 The direction problem at scale

With 124 pools to write, the labeled-list workflow (Section 11) hits a comprehension wall. Some response type names are misleading enough that even careful writers get the speaker direction wrong on first pass — especially the spy/intrigue category and the deal-pitch categories (WORK\_WITH\_US, RENEW\_DEAL, OPEN\_BORDERS\_OFFER, etc., where the name reads like an acceptance but the trigger is a proposal).

Mitigation: use the direction-aware label format from Section 11.4. The one-line direction note per pool turns the writing task from "interpret this response type name" into "write to this stated scenario." Significantly reduces the rewrite rate after in-game testing.

## 19.4 Iterate after shipping is the norm

With 124 pools (~500 lines), it is unrealistic to catch every direction issue before first ship. Plan to ship, play-test, find 3-8 direction mismatches across the playthrough, do targeted rewrites (Section 16), ship a point release. Repeat 2-3 times. By the third point release, most of the noticeable direction issues are resolved.

This is different from Update-based override mods, where the smaller scope means more issues catch in the writing phase and fewer survive to shipping. Don't apply the Update-mod mental model ("ship when the writing is done") to facelift work — the writing is only done after several rounds of play-test feedback.

## 19.5 The merge-pipeline pattern

A facelift mod's structural work is well-suited to scripted automation, separate from creative writing:

- Parse a labeled-list source draft (Section 11.4 format)
- Emit Row inserts for each language pool with the facelift Tag namespace
- Emit corresponding routing rows pointing at each pool with the % wildcard suffix
- Apply character normalization (Section 6.3) at emit time, not in the source draft
- Merge into the existing override XML, preserving all pre-existing legacy-namespace entries
- Validate: every facelift pool has matching routing, every routing row points at a populated pool, no duplicate Tag attributes, XML well-formed

The pipeline is mechanical and reusable across leaders. The writing is per-leader and irreplaceable. Keeping these two cleanly separated is the difference between a 30-minute targeted rewrite and a 3-hour structural rebuild when the pipeline misbehaves.

## 20. Closing Notes

The single most important habit is reading Database.log after every test. The log tells you exactly what failed. Guessing without it wastes hours; reading it saves them.

### Things worth repeating

- Always re-extract counts from the vanilla file for each leader. Static count tables go stale and carry typos.
- Avoid Delete and Replace in user mods. Add plain Rows to Diplomacy\_Responses. Both the blanket and targeted Delete forms misbehave in user mods.
- Use Update for existing legacy keys. Use Row only for keys you're certain don't exist anywhere, and for all facelift-namespace pool inserts.
- For BNW-era categories in the legacy namespace, default to Update — Firaxis quietly added many of these.
- Always check the vanilla routing file for your leader's exact warning prefix.
- Foreign-language vanilla files have the same Tag keys — use them when English files are missing.
- Delete the cache db after every rebuild before testing.



- Spell out contractions, rephrase possessives, use double-hyphen for dashes, strip non-ASCII letters.
- ModBuddy F7 is the real XML validator. Chat rendering color isn't.
- If a leader falls back to generic across the board, check Database.log for uniqueness errors first.
- `<Set Text="">` + loose text + `</Text>` = the malformed-tag bug. Search for that exact string to find every instance.
- Separate creative writing from XML — write in a labeled plain-text list first, then have an AI paste into the template.
- For facelift mods, include direction notes per label so speaker perspective is right the first time.
- Verify all three silent-failure properties before debugging anything else: Import into VFS = False, Action = OnModActivated → UpdateDatabase, AffectsSavedGames = 1.
- Saturate routed slots past vanilla counts to reduce perceived bleed (Section 15.1, 15.2). Don't try to saturate your way out of a direction-writing problem — direction is a writing fix, not a saturation fix.
- If a previously-working leader silently stops working, rebuild + cache clear before deeper debugging (Section 14, Pattern 9).
- When a line feels off but routing is correct, check direction (Section 14, Pattern 10) — the response type name may have led you to write the pool from the wrong speaker perspective.
- Expect to iterate after shipping a facelift mod. Direction issues only surface in-game across many diplomatic interactions. Use the targeted-rewrite workflow (Section 16) for each fix.

*Good luck modding.*