Branching & Workflow Guidelines

Base Branches

We maintain **three main branches**:

- **`main`** → Production
- **`SIT_KKDollar_iOS`** → SIT / QA testing
- **`Dev_KKDollar_iOS`** → Active development

All development work **must start from `Dev_KKDollar_iOS`** unless explicitly instructed otherwise.

1. Minor Bug Fix (No Functional Impact)

If you find a **minor bug** that does **not affect core functionality**:

Branching

- Create a branch from `Dev_KKDollar_iOS`

Branch naming format:
`Dev_KKDollar_iOS_<YourName>_<Date>`
Example: `Dev_KKDollar_iOS_Bhadresh_Feb10`

Workflow

1. Create branch from `Dev_KKDollar_iOS`
2. Fix the issue
3. Raise PR **only to `Dev_KKDollar_iOS`**
4. Merge after review

2. New Jira Task / Feature / Enhancement

For any **new Jira task**, feature, or enhancement:

Branching

- Always create a branch from `Dev_KKDollar_iOS`

Branch naming format:`<JIRA_TICKET>_<Short_Description>`

- (Use 1–2 meaningful words only)

Example
Jira: https://jira.luoma360s.com/browse/KKRPROD-6187
Title: Implement join_chat_table Protobuf Definition with Main Currency Support

Branch name: `KKRPROD-6187_Currency_Support`

Workflow

1. Create branch from `Dev_KKDollar_iOS`
2. Complete the task
3. Commit changes to **your task branch only**
4. Raise PR **only when instructed**

3. Working on Separate / Parallel Tasks

If you are working on a **separate task** or **long-running feature**:

- ❌ **Do NOT raise a PR to any branch**
- ✅ Commit changes **only to your task branch**
- PR will be created **only after confirmation**

This avoids accidental merges and keeps `Dev_KKDollar_iOS` stable.

Git Flow Summary

```
main
   ↑
SIT_KKDollar_iOS
   ↑
Dev_KKDollar_iOS
   ├── Dev_KKDollar_iOS_Bhadresh_Feb10   (minor bug fix)
   ├── KKRPROD-6187_Currency_Support     (Jira task)
   ├── KKRPROD-6201_Login_UI             (another task)
```

Rules to Remember

- ✅ Always branch from `Dev_KKDollar_iOS`
- ✅ Follow naming conventions strictly
- ❌ No direct commits to `Dev_KKDollar_iOS`, `SIT`, or `main`
- ❌ No PRs for unfinished or parallel tasks

🚀 KKR iOS – Selective Feature Release Strategy

🎯 Purpose

In **KKR iOS**, multiple Jira tasks are developed in parallel from `Dev_KKDollar_iOS`. However, not all completed tasks are released at the same time.

To ensure safe, controlled, and selective releases, we follow the structured release process described below.

All task branches must be created from:

Dev_KKDollar_iOS

🧩 Parallel Development Example

Ongoing Tasks

- ✅ `KKRPROD-6187_Currency_Support` (Ready for release)
- ✅ `KKRPROD-6190_PlayGame` (Ready for release)
- 🔄 `KKRPROD-6189_Language_Support` (Still in development – NOT ready)

All three tasks are being developed in parallel from `Dev_KKDollar_iOS`.

🎬 Release Scenario

We want to release only:

- ✅ `KKRPROD-6187_Currency_Support`
- ✅ `KKRPROD-6190_PlayGame`

We DO NOT want to release:

- ❌ `KKRPROD-6189_Language_Support`

📌 Step 1 — Create Release Branch

- Release Jira Ticket: **`PP-605`**
- Create release branch from: **Dev_KKDollar_iOS**
- Branch name must match Jira ticket:

PP-605

⚠ Release branch must always be created from **`Dev_KKDollar_iOS`**

📌 Step 2 — Merge Only Approved Tasks

Merge only selected task branches into `PP-605`

Example:

KKRPROD-6187_Currency_Support → PP-605

KKRPROD-6190_PlayGame → PP-605

Do NOT merge:

KKRPROD-6189_Language_Support

❗ Important Rules

- ❌ Do NOT merge entire `Dev_KKDollar_iOS`
- ❌ Do NOT merge unapproved task branches
- ❌ No direct merge from task branch → SIT or main
- ❌ No direct release from Dev → main
- ✅ Only approved tasks are included in release branch

🔁 Official Release Merge Flow

```
Selected Task Branches

       ↓

     PP-605

       ↓

  Dev_KKDollar_iOS   (sync back release changes)

       ↓

  SIT_KKDollar_iOS

       ↓

     main
```

📌 Step 3 — Sync Release Back to Dev (MANDATORY)

After testing is completed and release branch is finalized:

PP-605 → Dev_KKDollar_iOS

🔍 Why This Is Critical

Remember:
`KKRPROD-6189_Language_Support` is still ongoing.
It was created from `Dev_KKDollar_iOS`.

After release, developers will pull the latest code from `Dev_KKDollar_iOS`.

If release changes are NOT merged back:

- Ongoing tasks will miss released updates
- Future merge conflicts will increase
- Code mismatch between Dev and Production may happen
- Risk of regression issues

✅ This step ensures:

- Dev branch matches production state
- Ongoing tasks continue safely
- Stable code alignment

- Clean history management

📌 Step 4 — Merge to SIT

After syncing back to Dev:

Dev_KKDollar_iOS → SIT_KKDollar_iOS

- QA performs final verification
- Only release-approved code exists in SIT

📌 Step 5 — Production Release

After QA approval:

SIT_KKDollar_iOS → main

⚠ No direct merge from Dev to main.

🏷️ Step 6 — Create Release Tag (MANDATORY)

After merging into `main`, create a Git tag for the release.

Example:

If release version is: 7.1

Tag name must be: v7.1

Tagging Command Example:

git checkout main

git pull

git tag v7.1

git push origin v7.1

🎯 Why Tag Is Important

- Clear production version tracking
- Easy rollback capability
- Audit trail per release
- CI/CD automation reference

🔒 Strict Rules Summary

- Release branch must be created from `Dev_KKDollar_iOS`
- Release branch name = Jira ticket (e.g., `PP-605`)
- Only approved tasks are merged into release branch
- Release branch must be merged back into `Dev_KKDollar_iOS`
- No direct merge from task → SIT or main
- No direct release from Dev → main
- Every production release must have a Git tag

✅ Benefits of This Strategy

- Supports multiple parallel Jira tasks
- Allows selective feature release
- Prevents unfinished feature deployment
- Keeps Dev aligned with production
- Reduces future merge conflicts
- Provides clear audit trail per release
- Ensures controlled and traceable production deployments

📌 Final Release Checklist

Before marking release complete:

- Release branch created from Dev
- Only approved tasks merged
- Release synced back to Dev
- Merged into SIT
- QA approval completed
- Merged into main
- Git tag created (e.g., v7.1)
- CI/CD pipeline successful

Document Owner: iOS Team

Project: KKR iOS
Strategy Type: Selective Feature Release