

NIPR Managed Package - Development Guide

****For Hipten Developers****

Table of Contents

1. [Prerequisites](#prerequisites)
2. [Development Environment Setup](#development-environment-setup)
3. [Development Workflow](#development-workflow)
4. [Creating Package Versions](#creating-package-versions)
5. [Testing & Quality Assurance](#testing--quality-assurance)
6. [Deployment to Client Orgs](#deployment-to-client-orgs)
7. [Common Commands Reference](#common-commands-reference)
8. [Troubleshooting](#troubleshooting)

Prerequisites

Required Access

1. ****Permission Set****: `Hipten_Build_Managed_Packages`
 - Must be assigned by Hipten Administrator
 - Grants access to Dev Hub and package creation
2. ****GitHub Repository****: [https://github.com/stefan-nidzovic_hipten/NIPR](https://github.com/stefan-nidzovic_hipten/NIPR)
 - Request access from repository administrator
3. ****NIPR DEV Org****: <https://hipten2-dev-ed.develop.my.salesforce.com/>
 - Credentials: ****SuperAdmin Ht_Nipr**** (stored in LastPass)
 - This is the PRIMARY development org
4. ****Salesforce CLI****: Install from <https://developer.salesforce.com/tools/salesforcecli>

Development Environment Setup

Step 1: Authenticate to Dev Hub

```
# Login to Hipten Dev Hub (opens browser)
sf org login web --alias "HIPTEN DEV HUB" --set-default-dev-hub

# Verify authentication
sf org list --all
# Look for ■ icon next to HIPTEN DEV HUB
```

Step 2: Authenticate to NIPR DEV Org

```
# Login to NIPR DEV org (opens browser)
sf org login web --alias "NIPR DEV" --set-default

# Verify authentication
sf org display --target-org "NIPR DEV"
```

Step 3: Clone Repository

```
git clone https://github.com/stefan-nidzovic_hipten/NIPR.git
cd NIPR
```

Development Workflow

■ CRITICAL RULES - READ THIS FIRST ■

1. ■ ****ALL development happens in "NIPR DEV" org****
2. ■ ****Create feature branches**** from `main` for every new feature
3. ■ ****Pull metadata locally**** after development is complete
4. ■ ****Package versions are created from LOCAL repository**** - NOT from the org
5. ■ ****Test in QA org**** before releasing to clients

Step-by-Step Development Process

1. Create Feature Branch

```
# Ensure you're on main and up to date
git checkout main
git pull origin main

# Create new feature branch
git checkout -b feature/your-feature-name
```

2. Develop in NIPR DEV Org

```
# Open NIPR DEV org
sf org open --target-org "NIPR DEV"

# Make your changes in the org:
# - Create/modify Apex classes
# - Create/modify custom objects
# - Create/modify flows, reports, etc.
```

3. Pull Changes to Local Repository

After completing development in NIPR DEV org, pull your changes locally:

```
# Pull your changes from NIPR DEV org
sf project retrieve start --source-dir force-app --target-org "NIPR DEV"

# Verify what was retrieved
git status

# Review changes
git diff
```

4. Run Tests Locally

```
# Run all tests with coverage
sf apex run test --test-level RunLocalTests --code-coverage --result-format human --target-org "NIPR DEV"

# Must achieve minimum 75% code coverage
```

5. Commit and Push Changes

```
# Stage all changes
git add .

# Commit with descriptive message
git commit -m "feat: Add description of your feature"

# Push to remote
git push origin feature/your-feature-name
```

6. Create Pull Request

- Go to GitHub repository
- Create Pull Request from your feature branch to `main`
- Request code review from team lead
- Merge after approval

Creating Package Versions

Prerequisites

Before creating a package version:

1. ■ All changes merged to `main` branch
2. ■ Local repository synced with `main`
3. ■ All tests passing with $\geq 75\%$ coverage
4. ■ `sfdx-project.json` version number updated

Step 1: Update Version Number

Edit `sfdx-project.json`:

```
{
  "packageDirectories": [
    {
      "versionName": "ver 0.X",           // ← Update version name
      "versionNumber": "0.X.0.NEXT",     // ← Update version number
      "ancestorId": "04tXXXXXXXXXXXXXXX" // ← Previous Released version ID
    }
  ]
}
```

****Version Number Format**:** `major.minor.patch.NEXT`

- **Major**:** Breaking changes
- **Minor**:** New features, backward compatible
- **Patch**:** Bug fixes
- **NEXT**:** Auto-incremented by Salesforce

Step 2: Create Package Version

```
# Create package version with code coverage
sf package version create \
  --package "NIPR Integration" \
  --installation-key-bypass \
  --code-coverage \
  --wait 20 \
  --target-dev-hub "HIPTEN DEV HUB"

# This takes 10-20 minutes
# Note the Subscriber Package Version Id: 04tXXXXXXXXXXXXXXXXXX
```

Step 3: Promote to Released

```
# Promote package version to Released status
sf package version promote \
  --package "04tXXXXXXXXXXXXXXXXXX" \
  --target-dev-hub "HIPTEN DEV HUB" \
  --no-prompt
```

Step 4: Update ancestorId

After promoting, update `sfdx-project.json` with the new version as ancestor:

```
{
  "packageDirectories": [
    {
      "ancestorId": "04tXXXXXXXXXXXXXXXXXX" // ← New Released version ID
    }
  ]
}
```

Commit this change:

```
git add sfdx-project.json
git commit -m "chore: Update ancestorId to version 0.X.0-1"
git push origin main
```

Testing & Quality Assurance

QA Environment

****Hipten QA Org**:** `NIPR QA/UAT` (credentials stored in LastPass)

- Credentials: `[PLACEHOLDER - QA CREDENTIALS]` (stored in LastPass)

Step 1: Install Package in QA Org

```
# Authenticate to QA org (one-time setup)
sf org login web --alias "NIPR QA/UAT"

# Install the package version
sf package install \
  --package "04tXXXXXXXXXXXXXXXX" \
  --target-org "NIPR QA/UAT" \
  --wait 20
```

Step 2: Developer Unit Testing in QA

After package deployment to QA org, the ****developer**** performs unit testing:

- Test the specific feature they worked on
- Verify functionality works as expected
- Test edge cases related to the feature
- Validate any UI components or flows added

Once developer confirms the feature works, pass to Hipten QA team.

Step 3: QA Testing (Hipten QA Team)

Hipten QA team performs comprehensive testing in QA org:

- Full regression testing of all features
- Test integration with existing functionality
- Validate edge cases and error scenarios
- Check permission sets and sharing rules
- Verify reports and dashboards

Once QA passes, pass to Hipten UAT team.

Step 4: UAT Testing (Hipten UAT Team)

****UAT is performed in the SAME QA org**** after QA passes:

- Hipten internal stakeholders test the package
- Business users validate feature requirements meet business needs
- Sign-off required before client deployment

Deployment to Client Orgs

After UAT Approval

Once UAT passes in Hipten QA org, the ****SAME package version ID**** is deployed to client environments.

Client UAT Orgs

```
# Authenticate to client UAT org
sf org login web --alias "Client UAT Org Name"

# Install package
sf package install \
  --package "04tXXXXXXXXXXXXXXXX" \
  --target-org "Client UAT Org Name" \
  --wait 20
```

Client Production Orgs

****CRITICAL****: Only deploy to production **AFTER** client UAT approval.

```
# Authenticate to client production org
sf org login web --alias "Client Prod Org Name"

# Install/upgrade package
sf package install \
  --package "04tXXXXXXXXXXXXXXXX" \
  --target-org "Client Prod Org Name" \
  --wait 20 \
  --upgrade-type Mixed
```

Post-Deployment Steps

After deploying to client orgs:

1. Perform any required post-installation steps (see separate Post-Installation Guide)
2. Perform quick smoke test to verify installation success

Common Commands Reference

Quick Command Lookup

****Need help with these commands?**** Ask Claude:

> "Pull latest metadata from NIPR DEV and create a package version"

Claude has all these commands saved in the local CLAUDE.md file and can run them for you.

Deploy Metadata to NIPR DEV

```
sf project deploy start \  
  --source-dir force-app/main/default/classes \  
  --target-org "NIPR DEV" \  
  --wait 10
```

Retrieve Metadata from NIPR DEV

```
sf project retrieve start \  
  --source-dir force-app \  
  --target-org "NIPR DEV"
```

Run Tests

```
# Run all tests  
sf apex run test \  
  --test-level RunLocalTests \  
  --code-coverage \  
  --result-format human \  
  --target-org "NIPR DEV"  
  
# Run specific test class  
sf apex run test \  
  --class-names YourTestClass \  
  --result-format human \  
  --target-org "NIPR DEV"
```

List Package Versions

```
# List all package versions  
sf package version list \  
  --packages "NIPR Integration" \  
  --target-dev-hub "HIPTEN DEV HUB"  
  
# List only Released versions  
sf package version list \  
  --packages "NIPR Integration" \  
  --released
```

```
--released-only \  
--target-dev-hub "HIPTEN DEV HUB"
```

View Package Installation Status

```
sf package installed list --target-org "Org Alias"
```

Troubleshooting

Common Issues

Issue: "Not a Dev Hub" Error

****Solution**:** Ensure you have `Hipten_Build_Managed_Packages` permission set assigned, then re-authenticate:

```
sf org logout --target-org "HIPTEN DEV HUB" --no-prompt  
sf org login web --alias "HIPTEN DEV HUB" --set-default-dev-hub
```

Issue: Package Version Creation Fails with Coverage Error

****Solution**:** Package versions require code coverage to be promoted. Always use `--code-coverage` flag:

```
sf package version create \  
--package "NIPR Integration" \  
--installation-key-bypass \  
--code-coverage \  
--wait 20 \  
--target-dev-hub "HIPTEN DEV HUB"
```

Issue: "Can't create package version with version number 0.X.0.NEXT already exists"

****Solution**:** A Released version with that number already exists. Bump the version number in `sfdx-project.json`:

```
"versionNumber": "0.Y.0.NEXT" // Increment minor or major version
```

Issue: Metadata Type Not Supported in Package

****Solution**:** Some metadata types cannot be packaged (Groups, ApexTestSuites, etc.). These are already excluded in `.forceignore`. Check the file if new unsupported types are encountered.

Issue: Package Upgrade Fails in Target Org

****Solution****: Ensure `ancestorId` in `sfdx-project.json` points to the previous Released version. Check package version history:

```
sf package version list \  
  --packages "NIPR Integration" \  
  --target-dev-hub "HIPTEN DEV HUB"
```

Best Practices

1. ****Always develop in NIPR DEV org**** - Never develop locally or in scratch orgs
2. ****Pull metadata after every development session**** - Keep local repo in sync
3. ****Run tests before creating package versions**** - Avoid failed package builds
4. ****Use descriptive commit messages**** - Follow conventional commit format
5. ****Update version numbers correctly**** - Major.Minor.Patch.NEXT
6. ****Test in QA before client deployment**** - Never skip QA/UAT steps
7. ****Document breaking changes**** - Update release notes for major versions
8. ****Keep ancestorId updated**** - Ensures smooth package upgrades

Support

For questions or issues:

- **Development Questions****: Ask Claude (has all commands saved)
- **Repository Issues****: Contact repository administrator
- **Org Access****: Contact Hipten Administrator
- **Package Issues****: Contact Salesforce Partner Support

****Last Updated****: 2026-01-20

****Document Version****: 1.0

****Package Namespace****: `niprsync`

****Package Name****: `NIPR Integration`