STEP 40: File Reference Guide

Quick Reference for Executive Stakeholder Summary Workspace

Last Updated: December 2, 2025
Total Files: 9
Total Lines: 1,660+

Directory Structure

```
fyndr/nextjs_space/
├── prisma/
│   └── schema.prisma                                    [Database Schema]
├── lib/
│   ├── executive-summary/
│   │   └── composer.ts                                  [AI Summary Engine]
│   └── activity-types.ts                                [Activity Event Types]
├── app/
│   ├── api/dashboard/rfps/[id]/executive-summaries/
│   │   ├── route.ts                                     [List & Generate APIs]
│   │   └── [summaryId]/
│   │       ├── route.ts                                 [Get/Update/Delete APIs]
│   │       ├── autosave/route.ts                        [Autosave API]
│   │       ├── save-final/route.ts                      [Finalize API]
│   │       ├── clone/route.ts                           [Clone API]
│   │       └── pdf/route.ts                             [PDF Export API]
│   └── dashboard/rfps/[id]/executive-summary/
│       └── page.tsx                                     [UI Workspace]
└── STEP_40_EXECUTIVE_SUMMARY_WORKSPACE.md               [Documentation]
```

Database Schema

File: `prisma/schema.prisma`

ExecutiveSummaryDocument Model

Lines: 29
Location: Line ~XXX (search for “model ExecutiveSummaryDocument”)

Fields:

```
id              String    @id @default(cuid())
rfpId           String                                // Parent RFP
authorId        String                                // Creator user
title           String    @default("Executive Summary")
content         String    @db.Text                   // HTML content
tone            String    @default("professional")   // professional | persuasive
| analytical
audience        String    @default("executive")      // executive | technical | pro
curement
version         Int       @default(1)                // Version number
isOfficial      Boolean   @default(false)            // Official/final flag
clonedFromId    String?                               // Source if cloned
generatedAt     DateTime?                             // AI generation timestamp
autoSaveAt      DateTime?                             // Last autosave time
createdAt       DateTime  @default(now())
updatedAt       DateTime  @updatedAt
```

Relations:
- `rfp` → RFP (onDelete: Cascade)
- `author` → User

Indexes:
- `rfpId`
- `authorId`
- `isOfficial`

RFP Model Extension

Added Relation:

```
executiveSummaries  ExecutiveSummaryDocument[]
```

User Model Extension

Added Relation:

```
executiveSummaries  ExecutiveSummaryDocument[]
```

Backend Services

File: `lib/executive-summary/composer.ts`

Lines: 342

Types & Interfaces

ToneType

```
type ToneType = 'professional' | 'persuasive' | 'analytical';
```

AudienceType

```
type AudienceType = 'executive' | 'technical' | 'procurement';
```

SummaryGenerationOptions

```
interface SummaryGenerationOptions {
  rfpId: string;
  tone?: ToneType;
  audience?: AudienceType;
  userId: string;
}
```

Core Functions

1. generateExecutiveSummary()

```
async function generateExecutiveSummary(
  prisma: PrismaClient,
  options: SummaryGenerationOptions
): Promise<string>
```

- **Purpose:** Generate AI-powered executive summary
- **Returns:** HTML string with formatted summary
- **OpenAI Model:** GPT-4o
- **Error Handling:** Falls back to getFallbackSummary() on failure

2. gatherRFPContext()

```
async function gatherRFPContext(
  prisma: PrismaClient,
  rfp: RFPWithRelations
): Promise<Record<string, any>>
```

- **Purpose:** Aggregate RFP data for AI prompt
- **Aggregates:**
- RFP basics (title, description, budget, dates)
- Evaluation matrix criteria
- Opportunity score + breakdown
- Decision brief snapshot
- Supplier responses + top suppliers
- Scoring matrix snapshot
- **Returns:** Context object for prompt building

3. buildPrompt()

```
function buildPrompt(
  rfp: RFPWithRelations,
  context: Record<string, any>,
  tone: ToneType,
  audience: AudienceType
): string
```

- **Purpose:** Build structured prompt for OpenAI
- **Includes:**
- RFP details section
- Supplier engagement stats
- Opportunity score (if available)
- Top suppliers list
- Evaluation criteria
- Generation instructions
- **Returns:** Markdown-formatted prompt string

4. getSystemPrompt()

```
function getSystemPrompt(tone: ToneType, audience: AudienceType): string
```

- **Purpose:** Generate system-level instructions for OpenAI
- **Customizes:** Tone description and audience targeting
- **Returns:** System prompt with HTML formatting instructions

5. getToneTemperature()

```
function getToneTemperature(tone: ToneType): number
```

- **Purpose:** Map tone to OpenAI temperature parameter
- **Mapping:**
- `persuasive` : 0.8 (more creative)
- `analytical` : 0.3 (more deterministic)
- `professional` : 0.5 (balanced)
- **Returns:** Temperature value (0.0-1.0)

6. getFallbackSummary()

```
function getFallbackSummary(
  rfp: RFPWithRelations,
  context: Record<string, any>
): string
```

- **Purpose:** Generate summary when OpenAI fails
- **Structure:**
- Overview section
- Key metrics list
- Supplier engagement stats
- Opportunity score (if available)

- Top suppliers list
- Next steps
- **Returns:** HTML string with structured fallback content

7. sanitizeHTMLContent()

```
function sanitizeHTMLContent(content: string): string
```

- **Purpose:** Basic XSS prevention for HTML content
- **Removes:**
- `<script>` tags
- `<iframe>` tags
- `on*` event handlers
- **Note:** Basic sanitization; consider DOMPurify for production
- **Returns:** Sanitized HTML string

API Routes

Base Path: `/api/dashboard/rfps/[id]/executive-summaries`

---

1. List & Generate Summaries

File: `app/api/dashboard/rfps/[id]/executive-summaries/route.ts`
Lines: 165

GET /executive-summaries

Purpose: List all summaries for an RFP
Auth: Required (session)
Access: Buyer-only (RFP owner)
Query Params: None

Response:

```
{
  summaries: Array<{
    id: string;
    title: string;
    tone: string;
    audience: string;
    version: number;
    isOfficial: boolean;
    createdAt: DateTime;
    updatedAt: DateTime;
    author: {
      id: string;
      name: string;
      email: string;
    };
  }>;
}
```

Ordering: isOfficial DESC, version DESC, createdAt DESC

POST /executive-summaries/generate

Purpose: Generate new AI summary
Auth: Required (session)
Access: Buyer-only (RFP owner)

Request Body:

```
{
  tone?: 'professional' | 'persuasive' | 'analytical';  // default: 'professional'
  audience?: 'executive' | 'technical' | 'procurement';  // default: 'executive'
  title?: string;                                        // default: 'Executive Sum-
mary'
}
```

Response:

```
{
  summary: {
    id: string;
    rfpId: string;
    authorId: string;
    title: string;
    content: string;  // HTML
    tone: string;
    audience: string;
    version: number;
    isOfficial: false;
    generatedAt: DateTime;
    createdAt: DateTime;
    updatedAt: DateTime;
    author: { ... };
  };
}
```

Activity Log: EXECUTIVE_SUMMARY_GENERATED

---

2. Get/Update/Delete Specific Summary

File: `app/api/dashboard/rfps/[id]/executive-summaries/[summaryId]/route.ts`
Lines: 219

GET /executive-summaries/[summaryId]

Purpose: Get full details of specific summary
Auth: Required
Access: Buyer-only

Response:

```
{
  summary: {
    id: string;
    rfpId: string;
    authorId: string;
    title: string;
    content: string;  // HTML
    tone: string;
    audience: string;
    version: number;
    isOfficial: boolean;
    clonedFromId: string | null;
    generatedAt: DateTime | null;
    autoSaveAt: DateTime | null;
    createdAt: DateTime;
    updatedAt: DateTime;
    author: { ... };
  };
}
```

PATCH /executive-summaries/[summaryId]

Purpose: Update summary content (manual edit)
Auth: Required
Access: Buyer-only

Request Body:

```
{
  content: string;      // HTML content
  title?: string;       // Optional title update
  tone?: string;        // Optional tone update
  audience?: string;    // Optional audience update
}
```

Activity Log: EXECUTIVE_SUMMARY_EDITED

DELETE /executive-summaries/[summaryId]

Purpose: Delete summary
Auth: Required
Access: Buyer-only

Activity Log: EXECUTIVE_SUMMARY_DELETED

---

3. Autosave

File: `app/api/dashboard/rfps/[id]/executive-summaries/[summaryId]/autosave/route.ts`
Lines: 79

POST /executive-summaries/[summaryId]/autosave

Purpose: Save in-progress edits (lightweight, frequent)
Auth: Required
Access: Buyer-only

Request Body:

```
{
  content: string;  // HTML content
}
```

Behavior:
- Updates `content` field
- Updates `updatedAt` timestamp
- Updates `autoSaveAt` timestamp
- Does NOT change `isOfficial` or `version`
- Idempotent and safe for frequent calls

Response:

```
{
  success: true;
  summary: { ... };
}
```

Activity Log: None (intentional to reduce noise)

---

4. Save Final Version

File: `app/api/dashboard/rfps/[id]/executive-summaries/[summaryId]/save-final/route.ts`
Lines: 100

POST /executive-summaries/[summaryId]/save-final

Purpose: Mark summary as official/final version
Auth: Required
Access: Buyer-only

Request Body: None

Behavior:
- Sets `isOfficial = true`
- Updates `updatedAt` timestamp

Response:

```
{
  success: true;
  summary: { ... };
}
```

Activity Log: EXECUTIVE_SUMMARY_FINALIZED

---

5. Clone Summary

File: `app/api/dashboard/rfps/[id]/executive-summaries/[summaryId]/clone/route.ts`
Lines: 105

POST /executive-summaries/[summaryId]/clone

Purpose: Duplicate existing summary as new version
Auth: Required
Access: Buyer-only

Request Body: None (simplified implementation)

Behavior:
- Creates new ExecutiveSummaryDocument
- Copies `content`, `title`, `tone`, `audience`
- Sets `clonedFromId` to source summary ID
- Auto-increments `version` number
- Sets `isOfficial = false`

Response:

```
{
  success: true;
  summary: { ... };  // New cloned summary
}
```

Activity Log: EXECUTIVE_SUMMARY_CLONED

---

6. Export PDF

File: `app/api/dashboard/rfps/[id]/executive-summaries/[summaryId]/pdf/route.ts`
Lines: 163

GET /executive-summaries/[summaryId]/pdf

Purpose: Export summary as PDF document
Auth: Required
Access: Buyer-only

Query Params: None

Response:
- **Content-Type:** `text/html` (HTML document for PDF rendering)

- **Headers:**
- `Content-Disposition: attachment; filename="executive-summary-[rfpId]-v[version].html"`

Content Structure:
- RFP metadata (title, company, RFP ID)
- Summary metadata (version, tone, audience, author, date)
- Full HTML content from summary
- FYNDR branding footer
- Print-ready CSS styling

Activity Log: EXECUTIVE_SUMMARY_EXPORTED (format=pdf)

Note: Returns HTML for browser PDF rendering/printing. Could be enhanced with server-side PDF generation (Puppeteer, etc.)

---

7. Restore Version ❌ NOT IMPLEMENTED

Expected: `POST /executive-summaries/[summaryId]/restore`
Status: Endpoint missing

Workaround: Use clone endpoint + manual edit

UI Components

File: `app/dashboard/rfps/[id]/executive-summary/page.tsx`

Lines: 487

Main Component

```
export default async function ExecutiveSummaryPage({
  params
}: {
  params: { id: string }
})
```

Layout Structure

Header Bar
- Title: “Executive Stakeholder Summary”
- Tone selector (dropdown): professional, persuasive, analytical
- Audience selector (dropdown): executive, technical, procurement
- Buttons:
- “Generate Draft” → POST /executive-summaries/generate
- “Save Final Version” → POST /[summaryId]/save-final
- “Export PDF” → GET /[summaryId]/pdf
- Option3Indicator component

Left Pane: Version List
- List of summaries for RFP
- Grouped/ordered by version

- Each item shows:
- Title
- Tone pill/badge
- Version number
- isOfficial badge (if applicable)
- Updated timestamp
- Click to select and load into editor
- “New summary” button

Right Pane: Editor
- Rich text WYSIWYG editor (React Quill or similar)
- Formatting toolbar
- Autosave (debounced 3-5 seconds)
- Content stored as HTML

Status Bar
- “Autosaved at HH:MM” indicator
- “Official version” badge (if isOfficial)
- Tone and audience labels

State Management

```
const [tone, setTone] = useState<ToneType>('professional');
const [audience, setAudience] = useState<AudienceType>('executive');
const [selectedSummary, setSelectedSummary] = useState<Summary | null>(null);
const [summaries, setSummaries] = useState<Summary[]>([]);
const [content, setContent] = useState<string>('');
const [autoSaveStatus, setAutoSaveStatus] = useState<string>('');
```

Key Functions

loadSummaries()
- Fetches all summaries for RFP
- Updates summaries state
- Selects first official or latest summary

generateDraft()
- Calls POST /executive-summaries/generate
- Adds new summary to list
- Loads into editor

autosave()
- Debounced autosave function
- Calls POST /[summaryId]/autosave
- Updates status indicator

saveFinal()
- Calls POST /[summaryId]/save-final
- Updates isOfficial flag
- Shows success message

exportPDF()
- Opens GET /[summaryId]/pdf in new tab/window
- Browser handles download/print

cloneSummary()
- Calls POST /[summaryId]/clone
- Adds cloned summary to list
- Loads into editor

Activity Types

File: `lib/activity-types.ts`

Event Types

```
| "EXECUTIVE_SUMMARY_GENERATED"
| "EXECUTIVE_SUMMARY_EDITED"
| "EXECUTIVE_SUMMARY_FINALIZED"
| "EXECUTIVE_SUMMARY_CLONED"
| "EXECUTIVE_SUMMARY_DELETED"
| "EXECUTIVE_SUMMARY_EXPORTED"
```

Event Category

```
EXECUTIVE_SUMMARY: "EXECUTIVE_SUMMARY"
```

Friendly Names

```
EXECUTIVE_SUMMARY_GENERATED: "Executive Summary Generated"
EXECUTIVE_SUMMARY_EDITED: "Executive Summary Edited"
EXECUTIVE_SUMMARY_FINALIZED: "Executive Summary Finalized"
EXECUTIVE_SUMMARY_CLONED: "Executive Summary Cloned"
EXECUTIVE_SUMMARY_DELETED: "Executive Summary Deleted"
EXECUTIVE_SUMMARY_EXPORTED: "Executive Summary Exported"
```

Common Use Cases

1. Generate New Summary

```
// API Call
POST /api/dashboard/rfps/[rfpId]/executive-summaries/generate
Body: {
  tone: 'analytical',
  audience: 'executive',
  title: 'Q4 Marketing Platform RFP - Executive Brief'
}

// Returns new summary with AI-generated HTML content
```

2. Edit and Autosave

```
// User types in editor → debounced autosave triggers
POST /api/dashboard/rfps/[rfpId]/executive-summaries/[summaryId]/autosave
Body: {
  content: '<h2>Updated content...</h2>'
}

// Silent success, no UI disruption
```

3. Mark as Official

```
POST /api/dashboard/rfps/[rfpId]/executive-summaries/[summaryId]/save-final

// Sets isOfficial = true
// Logs EXECUTIVE_SUMMARY_FINALIZED event
```

4. Clone for New Version

```
POST /api/dashboard/rfps/[rfpId]/executive-summaries/[summaryId]/clone

// Creates new summary with incremented version
// Content copied from source
```

5. Export PDF

```
GET /api/dashboard/rfps/[rfpId]/executive-summaries/[summaryId]/pdf

// Returns HTML document
// User prints or saves as PDF from browser
```

Security Model

Authentication

- All endpoints require valid session (NextAuth)
- Unauthenticated requests → 401

Authorization

- RFP ownership check: `rfp.userId === session.user.id`
- Buyer-only: Suppliers cannot access any executive summary routes
- Company scoping: RFP must belong to user’s company

Data Validation

- HTML sanitization: `sanitizeHTMLContent()` removes scripts, iframes, event handlers
- Input validation: Tone and audience validated against allowed types
- Database constraints: Foreign keys enforce referential integrity

---

Performance Considerations

Database

- Indexes on: `rfpId`, `authorId`, `isOfficial`
- Cascade delete: Summaries deleted when RFP deleted
- Efficient ordering: isOfficial DESC, version DESC

API

- Autosave debounced to reduce requests
- Lightweight autosave endpoint (no activity log)
- OpenAI timeout handling with fallback

UI

- Debounced autosave (3-5 seconds)
- Conditional rendering of large lists
- Status indicators prevent redundant saves

Error Handling

OpenAI Failures

- Catch blocks in `generateExecutiveSummary()`
- Falls back to `getFallbackSummary()`
- Console logs error for debugging

API Errors

- Try-catch blocks in all route handlers
- Appropriate HTTP status codes (401, 403, 404, 500)
- User-friendly error messages

UI Errors

- Toast/alert notifications for user
- Error states in components
- Graceful degradation

Testing Checklist

Functional Tests

- [ ] Generate summary with each tone/audience combination
- [ ] Edit summary and verify autosave triggers
- [ ] Mark summary as official
- [ ] Clone summary and verify new version created
- [ ] Export PDF and verify content/formatting
- [ ] Create multiple versions and verify list ordering

- [ ] Test empty state (no summaries yet)

Security Tests

- [ ] Attempt access as supplier → 403
- [ ] Attempt cross-company access → 403/404
- [ ] Attempt access without authentication → 401
- [ ] Verify XSS protection with malicious HTML

Performance Tests

- [ ] Load page with 10+ summaries
- [ ] Test autosave frequency and responsiveness
- [ ] Test OpenAI generation time
- [ ] Test PDF export with large content

Quick Commands

Database

```
# Sync schema
cd /home/ubuntu/fyndr/nextjs_space
npx prisma db push

# Check migration status
npx prisma migrate status

# View data
npx prisma studio
```

Build & Test

```
# Build project
npm run build

# Type check
npx tsc --noEmit

# Run dev server
npm run dev
```

Git

```
# View modified files
git status

# Commit STEP 40
git add -A
git commit -m "STEP 40: Complete Executive Stakeholder Summary Workspace"
```

External Dependencies

npm Packages

- `openai` - OpenAI API client
- `react-quill` or similar - Rich text editor (verify in package.json)
- `@prisma/client` - Database ORM
- `next-auth` - Authentication

Environment Variables

- `OPENAI_API_KEY` - Required for AI generation
- `DATABASE_URL` - Prisma connection string
- `NEXTAUTH_SECRET` - NextAuth encryption

Related Documentation

- **STEP 40 Completion Report:** `/home/ubuntu/fyndr/STEP_40_COMPLETION_REPORT.md`
- **STEP 40 Executive Summary:** `/home/ubuntu/fyndr/STEP_40_EXECUTIVE_SUMMARY.md`
- **Technical Documentation:** `/home/ubuntu/fyndr/nextjs_space/STEP_40_EXECUTIVE_SUMMARY_WORKSPACE.md`
- **Prisma Schema:** `/home/ubuntu/fyndr/nextjs_space/prisma/schema.prisma`

Generated: December 2, 2025
For: FYNDR Platform - STEP 40 Implementation
Quick reference for developers and maintainers