Contents

SECTION 30: DYNAMIC PROVIDER REGISTRY + xAI/GROK (v3.7.0)

> **NEW in v3.7.0**: Database-driven model registry with automatic sync and xAI/Grok integration.

30.1 Complete Provider & Model Registry

External AI Providers (21+)

Provider ID	Display Name	API Base	Auth Type
`anthropic`	Anthropic	api.anthropic.com	API Key
`openai`	OpenAI	api.openai.com	API Key
`google`	Google (Gemini)	generativelanguage.googleapis.com	API Key
`xai`	xAI (Grok)	api.x.ai	API Key
`mistral`	Mistral AI	api.mistral.ai	API Key
`cohere`	Cohere	api.cohere.ai	API Key
`perplexity`	Perplexity	api.perplexity.ai	API Key
`deepseek`	DeepSeek	api.deepseek.com	API Key
`together`	Together AI	api.together.xyz	API Key
`fireworks`	Fireworks AI	api.fireworks.ai	API Key
`groq`	Groq	api.groq.com	API Key
`replicate`	Replicate	api.replicate.com	API Key
`huggingface`	Hugging Face	api-inference.huggingface.co	API Key
`bedrock`	AWS Bedrock	bedrock-runtime.amazonaws.com	IAM
`azure_openai`	Azure OpenAI	*.openai.azure.com	API Key
`vertex_ai`	Google Vertex AI	*.aiplatform.googleapis.com	Service Account

Provider ID	Display Name	API Base	Auth Type

30.2 xAI/Grok Models (10 Models)

Model ID	Display Name	Context	Input $/1M	Output $/1M	Capabilities
grok-3	Grok 3	131K	$3.00	$15.00	Flagship, real-time info
grok-3-fast	Grok 3 Fast	131K	$1.00	$5.00	Speed-optimized
grok-3-mini	Grok 3 Mini	131K	$0.30	$1.50	Cost-effective
grok-2	Grok 2	131K	$2.00	$10.00	Previous generation
grok-2-vision	Grok 2 Vision	32K	$2.00	$10.00	Image understanding
grok-2-mini	Grok 2 Mini	131K	$0.20	$1.00	Budget option
grok-coder	Grok Coder	131K	$1.50	$7.50	Code generation
grok-analyst	Grok Analyst	131K	$2.00	$10.00	Data analysis
grok-embed	Grok Embed	8K	$0.10	-	Text embeddings
grok-realtime	Grok Realtime	32K	$5.00	$20.00	Voice/streaming

xAI Provider Configuration

```
// packages/shared/src/providers/xai.ts

export const XAI_PROVIDER_CONFIG = {
  id: 'xai',
  displayName: 'xAI (Grok)',
  apiBase: 'https://api.x.ai/v1',
  authType: 'api_key',
  authHeader: 'Authorization',
  authPrefix: 'Bearer ',

  models: [
    {
      id: 'grok-3',
      displayName: 'Grok 3',
      contextWindow: 131072,
      maxOutputTokens: 8192,
      supportedModalities: ['text'],
      pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
      capabilities: ['chat', 'reasoning', 'analysis', 'realtime_info'],
      isNovel: false,
    },
    {
```

```
    id: 'grok-3-fast',
    displayName: 'Grok 3 Fast',
    contextWindow: 131072,
    maxOutputTokens: 8192,
    supportedModalities: ['text'],
    pricing: { inputPer1M: 1.00, outputPer1M: 5.00 },
    capabilities: ['chat', 'fast_response'],
    isNovel: false,
  },
  {
    id: 'grok-3-mini',
    displayName: 'Grok 3 Mini',
    contextWindow: 131072,
    maxOutputTokens: 4096,
    supportedModalities: ['text'],
    pricing: { inputPer1M: 0.30, outputPer1M: 1.50 },
    capabilities: ['chat', 'cost_effective'],
    isNovel: false,
  },
  {
    id: 'grok-2',
    displayName: 'Grok 2',
    contextWindow: 131072,
    maxOutputTokens: 8192,
    supportedModalities: ['text'],
    pricing: { inputPer1M: 2.00, outputPer1M: 10.00 },
    capabilities: ['chat', 'reasoning'],
    isNovel: false,
  },
  {
    id: 'grok-2-vision',
    displayName: 'Grok 2 Vision',
    contextWindow: 32768,
    maxOutputTokens: 4096,
    supportedModalities: ['text', 'image'],
    pricing: { inputPer1M: 2.00, outputPer1M: 10.00 },
    capabilities: ['chat', 'vision', 'image_analysis'],
    isNovel: true,
  },
  {
    id: 'grok-2-mini',
    displayName: 'Grok 2 Mini',
    contextWindow: 131072,
    maxOutputTokens: 4096,
    supportedModalities: ['text'],
    pricing: { inputPer1M: 0.20, outputPer1M: 1.00 },
    capabilities: ['chat'],
    isNovel: false,
    },
    {
      id: 'grok-coder',
      displayName: 'Grok Coder',
      contextWindow: 131072,
      maxOutputTokens: 8192,
      supportedModalities: ['text'],
      pricing: { inputPer1M: 1.50, outputPer1M: 7.50 },
      capabilities: ['chat', 'code_generation', 'code_review'],
      isNovel: true,
    },
    {
      id: 'grok-analyst',
      displayName: 'Grok Analyst',
      contextWindow: 131072,
      maxOutputTokens: 8192,
      supportedModalities: ['text'],
      pricing: { inputPer1M: 2.00, outputPer1M: 10.00 },
      capabilities: ['chat', 'data_analysis', 'insights'],
      isNovel: true,
    },
    {
      id: 'grok-embed',
      displayName: 'Grok Embed',
      contextWindow: 8192,
      maxOutputTokens: 0,
      supportedModalities: ['text'],
      pricing: { inputPer1M: 0.10, outputPer1M: 0 },
      capabilities: ['embeddings'],
      isNovel: false,
    },
    {
      id: 'grok-realtime',
      displayName: 'Grok Realtime',
      contextWindow: 32768,
      maxOutputTokens: 4096,
      supportedModalities: ['text', 'audio'],
      pricing: { inputPer1M: 5.00, outputPer1M: 20.00 },
      capabilities: ['chat', 'voice', 'streaming', 'realtime'],
      isNovel: true,
    },
  ],
};
```

30.3 Complete Model Registry

All External Models (60+)

```
// packages/shared/src/models/registry.ts

export const MODEL_REGISTRY: ModelDefinition[] = [
  //
  // ANTHROPIC MODELS
  //
  {
    id: 'claude-4-opus',
    providerId: 'anthropic',
    displayName: 'Claude 4 Opus',
    contextWindow: 200000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 15.00, outputPer1M: 75.00 },
    capabilities: ['chat', 'reasoning', 'analysis', 'code', 'vision'],
    isNovel: false,
    category: 'flagship',
  },
  {
    id: 'claude-4-sonnet',
    providerId: 'anthropic',
    displayName: 'Claude 4 Sonnet',
    contextWindow: 200000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
    capabilities: ['chat', 'reasoning', 'analysis', 'code', 'vision'],
    isNovel: false,
    category: 'balanced',
  },
  {
    id: 'claude-3.5-haiku',
    providerId: 'anthropic',
    displayName: 'Claude 3.5 Haiku',
    contextWindow: 200000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.25, outputPer1M: 1.25 },
    capabilities: ['chat', 'code'],
    isNovel: false,
    category: 'economy',
  },

  //
  // OPENAI MODELS
  //
  {
    id: 'gpt-4o',
    providerId: 'openai',
    displayName: 'GPT-4o',
    contextWindow: 128000,
    maxOutputTokens: 16384,
    pricing: { inputPer1M: 2.50, outputPer1M: 10.00 },
    capabilities: ['chat', 'reasoning', 'vision', 'audio'],
    isNovel: false,
    category: 'flagship',
  },
  {
    id: 'gpt-4o-mini',
    providerId: 'openai',
    displayName: 'GPT-4o Mini',
    contextWindow: 128000,
    maxOutputTokens: 16384,
    pricing: { inputPer1M: 0.15, outputPer1M: 0.60 },
    capabilities: ['chat', 'vision'],
    isNovel: false,
    category: 'economy',
  },
  {
    id: 'o1',
    providerId: 'openai',
    displayName: 'o1 Reasoning',
    contextWindow: 200000,
    maxOutputTokens: 100000,
    pricing: { inputPer1M: 15.00, outputPer1M: 60.00 },
    capabilities: ['reasoning', 'analysis', 'math', 'code'],
    isNovel: false,
    category: 'specialized',
  },
  {
    id: 'o1-pro',
    providerId: 'openai',
    displayName: 'o1 Pro',
    contextWindow: 200000,
    maxOutputTokens: 100000,
    pricing: { inputPer1M: 150.00, outputPer1M: 600.00 },
    capabilities: ['reasoning', 'analysis', 'math', 'code', 'extended_thinking'],
    isNovel: true,
    category: 'novel',
  },
  {
    id: 'gpt-4o-realtime',
    providerId: 'openai',
    displayName: 'GPT-4o Realtime',
    contextWindow: 128000,
    maxOutputTokens: 4096,
    pricing: { inputPer1M: 5.00, outputPer1M: 20.00 },
    capabilities: ['voice', 'streaming', 'realtime'],
    isNovel: true,
    category: 'novel',
  },

  //
  // GOOGLE MODELS
  //
  {
    id: 'gemini-2.0-pro',
    providerId: 'google',
    displayName: 'Gemini 2.0 Pro',
    contextWindow: 2000000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 1.25, outputPer1M: 5.00 },
    capabilities: ['chat', 'reasoning', 'vision', 'code'],
    isNovel: false,
    category: 'flagship',
  },
  {
    id: 'gemini-2.0-flash',
    providerId: 'google',
    displayName: 'Gemini 2.0 Flash',
    contextWindow: 1000000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.075, outputPer1M: 0.30 },
    capabilities: ['chat', 'vision', 'fast'],
    isNovel: false,
    category: 'balanced',
  },
  {
    id: 'gemini-2.0-ultra',
    providerId: 'google',
    displayName: 'Gemini 2.0 Ultra',
    contextWindow: 2000000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 5.00, outputPer1M: 15.00 },
    capabilities: ['chat', 'reasoning', 'vision', 'multimodal'],
    isNovel: true,
    category: 'novel',
  },
  {
    id: 'gemini-2.0-pro-exp',
    providerId: 'google',
    displayName: 'Gemini Pro Experimental',
    contextWindow: 10000000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 2.50, outputPer1M: 10.00 },
    capabilities: ['chat', 'reasoning', 'massive_context'],
    isNovel: true,
    category: 'novel',
  },

  //
  // XAI/GROK MODELS (from 30.2)
  //
  {
    id: 'grok-3',
    providerId: 'xai',
    displayName: 'Grok 3',
    contextWindow: 131072,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
    capabilities: ['chat', 'reasoning', 'realtime_info'],
    isNovel: false,
    category: 'flagship',
  },
  {
    id: 'grok-3-fast',
    providerId: 'xai',
    displayName: 'Grok 3 Fast',
    contextWindow: 131072,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 1.00, outputPer1M: 5.00 },
    capabilities: ['chat', 'fast'],
    isNovel: false,
    category: 'balanced',
  },
  {
    id: 'grok-3-mini',
    providerId: 'xai',
    displayName: 'Grok 3 Mini',
    contextWindow: 131072,
    maxOutputTokens: 4096,
    pricing: { inputPer1M: 0.30, outputPer1M: 1.50 },
    capabilities: ['chat'],
    isNovel: false,
    category: 'economy',
  },

  //
  // DEEPSEEK MODELS
  //
  {
    id: 'deepseek-v3',
    providerId: 'deepseek',
    displayName: 'DeepSeek V3',
    contextWindow: 64000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.14, outputPer1M: 0.28 },
    capabilities: ['chat', 'code', 'reasoning'],
    isNovel: false,
    category: 'economy',
  },
  {
    id: 'deepseek-r1',
    providerId: 'deepseek',
    displayName: 'DeepSeek R1',
    contextWindow: 64000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.55, outputPer1M: 2.19 },
    capabilities: ['reasoning', 'chain_of_thought', 'analysis'],
    isNovel: true,
    category: 'novel',
  },

  //
  // MISTRAL MODELS
  //
  {
    id: 'mistral-large-2',
    providerId: 'mistral',
    displayName: 'Mistral Large 2',
    contextWindow: 128000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 2.00, outputPer1M: 6.00 },
    capabilities: ['chat', 'reasoning', 'multilingual'],
    isNovel: false,
    category: 'specialized',
  },
  {
    id: 'codestral-latest',
    providerId: 'mistral',
    displayName: 'Codestral',
    contextWindow: 32000,
    maxOutputTokens: 8192,
    pricing: { inputPer1M: 0.30, outputPer1M: 0.90 },
    capabilities: ['code', 'code_generation', 'code_review'],
    isNovel: false,
    category: 'specialized',
  },
//
// PERPLEXITY MODELS
//
{
  id: 'perplexity-sonar-pro',
  providerId: 'perplexity',
  displayName: 'Sonar Pro',
  contextWindow: 128000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 3.00, outputPer1M: 15.00 },
  capabilities: ['search', 'chat', 'citations', 'realtime_info'],
  isNovel: false,
  category: 'specialized',
},

//
// NOVEL/EXPERIMENTAL MODELS
//
{
  id: 'claude-4-opus-agents',
  providerId: 'anthropic',
  displayName: 'Claude Opus Agents',
  contextWindow: 200000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 15.00, outputPer1M: 75.00 },
  capabilities: ['chat', 'tool_use', 'computer_use', 'agents'],
  isNovel: true,
  category: 'novel',
},
{
  id: 'qwen-2.5-coder',
  providerId: 'together',
  displayName: 'Qwen 2.5 Coder',
  contextWindow: 128000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 0.30, outputPer1M: 0.90 },
  capabilities: ['code', 'code_generation'],
  isNovel: true,
  category: 'novel',
},
{
  id: 'llama-3.3-70b',
  providerId: 'together',
  displayName: 'Llama 3.3 70B',
  contextWindow: 128000,
  maxOutputTokens: 8192,
  pricing: { inputPer1M: 0.88, outputPer1M: 0.88 },
  capabilities: ['chat', 'reasoning', 'open_weights'],
    isNovel: true,
    category: 'novel',
  },
  {
    id: 'phi-4',
    providerId: 'azure_openai',
    displayName: 'Phi-4',
    contextWindow: 16000,
    maxOutputTokens: 4096,
    pricing: { inputPer1M: 0.07, outputPer1M: 0.14 },
    capabilities: ['chat', 'reasoning', 'efficient'],
    isNovel: true,
    category: 'novel',
  },
];
```

30.4 Provider Sync Lambda

```
// packages/lambdas/src/handlers/admin/sync-providers.ts

import { ScheduledHandler } from 'aws-lambda';
import { Pool } from 'pg';
import { MODEL_REGISTRY } from '@radiant/shared';
import { createLogger } from '@radiant/shared';

const pool = new Pool({ connectionString: process.env.DATABASE_URL });
const logger = createLogger('provider-sync');

export const handler: ScheduledHandler = async () => {
  logger.info('Starting provider/model sync');

  const client = await pool.connect();

  try {
    await client.query('BEGIN');

    // Sync all models from registry
    for (const model of MODEL_REGISTRY) {
      await client.query(`
        INSERT INTO models (
          id, provider_id, display_name, model_type, context_window,
          max_output_tokens, pricing, capabilities, is_novel, category,
          is_enabled, updated_at
        ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8, $9, $10, TRUE, NOW())
        ON CONFLICT (id) DO UPDATE SET
          display_name = EXCLUDED.display_name,
          context_window = EXCLUDED.context_window,
          max_output_tokens = EXCLUDED.max_output_tokens,
          capabilities = EXCLUDED.capabilities,
          is_novel = EXCLUDED.is_novel,
          category = EXCLUDED.category,
          updated_at = NOW()
        -- Note: pricing is NOT updated to preserve admin overrides
      `, [
        model.id,
        model.providerId,
        model.displayName,
        'chat',
        model.contextWindow,
        model.maxOutputTokens,
        JSON.stringify(model.pricing),
        JSON.stringify(model.capabilities),
        model.isNovel || false,
        model.category || 'general',
      ]);
    }

    await client.query('COMMIT');
    logger.info(`Synced ${MODEL_REGISTRY.length} models`);

  } catch (error) {
    await client.query('ROLLBACK');
    logger.error('Sync failed', { error });
    throw error;
  } finally {
    client.release();
  }
};
```