

- Site Allocation Calculation Documentation
 - Overview
 - Table of Contents
 - 1. Base Requirement Calculation
 - Location in Code
 - Calculation Formulas by Program
 - Paint Program (HSP)
 - Lighting (EEE) Program
 - Solvents & Pesticides Programs (HSP)
 - Code Implementation
 - Quick Reference: Population Thresholds
 - 2. Compliance Calculation
 - Location in Code
 - Calculation Process
 - Compliance Result Structure
 - Site Counting Rules
 - 3. Tool A: Direct Service Offset
 - Location in Code
 - Purpose
 - Calculation Formula
 - Code Implementation
 - Detailed Examples
 - Example 1: 10% Global Reduction for Paint Program
 - Example 2: 25% Reduction for Lighting Program
 - Example 3: Per-Community Override
 - Features
 - Important Notes
 - 4. Tool B: Event Application
 - Location in Code
 - Purpose
 - Key Constraints
 - Workflow
 - Statistics Tracked
 - Detailed Examples
 - Example 1: Calculating Maximum Events Allowed
 - Example 2: Applying Events to Communities with Shortfalls
 - Example 3: Apply All Respecting 35% Cap

- 5. Tool C: Adjacent Community Reallocation
 - Location in Code
 - Purpose
 - Eligibility Rules
 - Sites that CAN be reallocated:
 - Sites that CANNOT be reallocated:
 - Calculation Process
 - Reallocation Rules by Program
 - EEE (Lighting) Program:
 - HSP Programs (Paint, Solvents, Pesticides):
 - Percentage Limits
 - Detailed Examples
 - Example 1: Identifying Eligible Excess
 - Example 2: Reallocation to Adjacent Community
 - Example 3: Program-Specific Reallocation Rules
- 6. Data Models
 - Municipality (Community)
 - Collection Site
 - Reallocation
 - Operator Types
- 7. Business Rules & Constraints
 - Site Counting Rules
 - Offset Limits
 - Reallocation Eligibility
 - Program-Specific Rules
 - Minimum Requirements
- Summary Flowchart
- 8. Complete Calculation Examples
 - End-to-End Example: Paint Program Compliance for York Region
 - Step 1: Calculate Base Requirements
 - Step 2: Apply Tool A Direct Service Offset (10% Global)
 - Step 3: Count Actual Sites (Excluding Events)
 - Step 4: Calculate Shortfall/Excess
 - Step 5: Apply Tool B Events
 - Step 6: Apply Tool C Adjacent Reallocation
 - Final Compliance Summary
 - Quick Reference: Calculation Cheat Sheet
 - Base Requirement Formulas

- [Key Limits](#)
- [File References](#)

Site Allocation Calculation Documentation

Overview

This document provides comprehensive documentation on how site allocation calculations are performed in the ArcGIS Compliance Tool. The system calculates the required number of collection sites for each community based on population and program type, then provides tools to manage compliance through offsets, event applications, and adjacent community reallocations.

Table of Contents

1. [Base Requirement Calculation](#)
 2. [Compliance Calculation](#)
 3. [Tool A: Direct Service Offset](#)
 4. [Tool B: Event Application](#)
 5. [Tool C: Adjacent Community Reallocation](#)
 6. [Data Models](#)
 7. [Business Rules & Constraints](#)
 8. [Complete Calculation Examples](#)
-

1. Base Requirement Calculation

The core calculation determines how many collection sites a community requires based on its **population** and the **program type**.

Location in Code

- **Primary:** `lib/compliance.ts` → `calculateRequirements()`
- **Also implemented in:** Each tool component (`tool-a-direct-service-offset.tsx`, `tool-b-event-application.tsx`, `tool-c-adjacent-reallocation.tsx`)

Calculation Formulas by Program

Paint Program (HSP)

Population Range Divisor	Formula	
-----	-----	----

< 1,000	0 sites required	N/A
1,000 – 4,999	1 site required (flat)	N/A
5,000 – 500,000	ceil(population / 40,000)	
40,000		
> 500,000	13 + ceil((population – 500,000) / 150,000)	
150,000		

Step-by-Step Example Calculations (Paint):

Population	Range Check	Formula Applied	Calculation	Result
500	< 1,000	Return 0	N/A	0 sites
2,500	1,000 - 4,999	Return 1	N/A	1 site
40,000	5,000 - 500,000	ceil(40000 / 40000)	ceil(1.0)	1 site
50,000	5,000 - 500,000	ceil(50000 / 40000)	ceil(1.25)	2 sites
120,000	5,000 - 500,000	ceil(120000 / 40000)	ceil(3.0)	3 sites
200,000	5,000 - 500,000	ceil(200000 / 40000)	ceil(5.0)	5 sites
500,000	5,000 - 500,000	ceil(500000 / 40000)	ceil(12.5)	13 sites

Population	Range Check	Formula Applied	Calculation	Result
600,000	> 500,000	13 + ceil(100000 / 150000)	13 + ceil(0.67)	14 sites
750,000	> 500,000	13 + ceil(250000 / 150000)	13 + ceil(1.67)	15 sites
1,000,000	> 500,000	13 + ceil(500000 / 150000)	13 + ceil(3.33)	17 sites
2,800,000	> 500,000	13 + ceil(2300000 / 150000)	13 + ceil(15.33)	29 sites

Lighting (EEE) Program

Population Range	Divisor	Formula	
< 1,000		0 sites required	N/A
1,000 – 500,000		ceil(population / 15,000)	
> 500,000		34 + ceil((population – 500,000) / 50,000)	

Step-by-Step Example Calculations (Lighting):

Population	Range Check	Formula Applied	Calculation	Result
500	< 1,000	Return 0	N/A	0 sites
1,000	1,000 - 500,000	ceil(1000 / 15000)	ceil(0.067)	1 site
15,000	1,000 - 500,000	ceil(15000 / 15000)	ceil(1.0)	1 site
50,000	1,000 - 500,000	ceil(50000 / 15000)	ceil(3.33)	4 sites
100,000	1,000 - 500,000	ceil(100000 / 15000)	ceil(6.67)	7 sites

Population	Range Check	Formula Applied	Calculation	Result
200,000	1,000 - 500,000	$\text{ceil}(200000 / 15000)$	$\text{ceil}(13.33)$	14 sites
500,000	1,000 - 500,000	$\text{ceil}(500000 / 15000)$	$\text{ceil}(33.33)$	34 sites
550,000	> 500,000	$34 + \text{ceil}(50000 / 50000)$	$34 + \text{ceil}(1.0)$	35 sites
750,000	> 500,000	$34 + \text{ceil}(250000 / 50000)$	$34 + \text{ceil}(5.0)$	39 sites
1,000,000	> 500,000	$34 + \text{ceil}(500000 / 50000)$	$34 + \text{ceil}(10.0)$	44 sites
2,800,000	> 500,000	$34 + \text{ceil}(2300000 / 50000)$	$34 + \text{ceil}(46.0)$	80 sites

Solvents & Pesticides Programs (HSP)

Population Range Divisor	Formula	
----- ----- ---		
< 1,000 N/A	0 sites required	
1,000 – 9,999 N/A	1 site required (flat)	
10,000 – 500,000 250,000	$\text{ceil}(\text{population} / 250,000)$	
> 500,000 300,000	$2 + \text{ceil}((\text{population} - 500,000) / 300,000)$	

Step-by-Step Example Calculations (Solvents/Pesticides):

Population	Range Check	Formula Applied	Calculation	Result
500	< 1,000	Return 0	N/A	0 sites
5,000	1,000 - 9,999	Return 1	N/A	1 site
10,000	10,000 - 500,000	$\text{ceil}(10000 / 250000)$	$\text{ceil}(0.04)$	1 site

Population	Range Check	Formula Applied	Calculation	Result
50,000	10,000 - 500,000	$\text{ceil}(50000 / 250000)$	$\text{ceil}(0.2)$	1 site
250,000	10,000 - 500,000	$\text{ceil}(250000 / 250000)$	$\text{ceil}(1.0)$	1 site
300,000	10,000 - 500,000	$\text{ceil}(300000 / 250000)$	$\text{ceil}(1.2)$	2 sites
500,000	10,000 - 500,000	$\text{ceil}(500000 / 250000)$	$\text{ceil}(2.0)$	2 sites
600,000	> 500,000	$2 + \text{ceil}(100000 / 300000)$	$2 + \text{ceil}(0.33)$	3 sites
900,000	> 500,000	$2 + \text{ceil}(400000 / 300000)$	$2 + \text{ceil}(1.33)$	4 sites
1,000,000	> 500,000	$2 + \text{ceil}(500000 / 300000)$	$2 + \text{ceil}(1.67)$	4 sites
2,800,000	> 500,000	$2 + \text{ceil}(2300000 / 300000)$	$2 + \text{ceil}(7.67)$	10 sites

Code Implementation

```
// From lib/compliance.ts
export function calculateRequirements(population: number, program: string):
number {
  switch (program) {
    case "Paint":
      if (population >= 5000 && population <= 500000) {
        return Math.ceil(population / 40000)
      } else if (population > 500000) {
        return 13 + Math.ceil((population - 500000) / 150000)
      }
      return population >= 1000 ? 1 : 0

    case "Solvents":
    case "Pesticides":
      if (population >= 10000 && population <= 500000) {
        return Math.ceil(population / 250000)
      } else if (population > 500000) {
        return 2 + Math.ceil((population - 500000) / 300000)
      }
      return population >= 1000 ? 1 : 0
  }
}
```

```

    case "Lighting":
      if (population >= 1000 && population <= 500000) {
        return Math.ceil(population / 15000)
      } else if (population > 500000) {
        return 34 + Math.ceil((population - 500000) / 50000)
      }
      return 0

    default:
      return 0
  }
}

```

Quick Reference: Population Thresholds

Program	Min Pop for 1 Site	Pop per Site (Standard)	Pop per Site (Large)	Large Pop Threshold
Paint	1,000	40,000	150,000	500,000
Lighting	1,000	15,000	50,000	500,000
Solvents	1,000	250,000	300,000	500,000
Pesticides	1,000	250,000	300,000	500,000

2. Compliance Calculation

Compliance is determined by comparing the **required** number of sites against the **actual** number of sites in each community.

Location in Code

- `lib/compliance.ts` → `calculateCompliance()`
- `components/compliance-analysis.tsx`

Calculation Process

1. For each municipality and program:

- Calculate required sites using the formulas above
- Count actual sites that match:
 - Same municipality
 - Same program
 - Status is "Active" or "Scheduled"

2. Determine compliance status:

```
const actual = municipalitySites.length
const shortfall = Math.max(0, required - actual)
const excess = Math.max(0, actual - required)
const complianceRate = required > 0 ? (actual / required) * 100 : 100

let status: "compliant" | "shortfall" | "excess" = "compliant"
if (shortfall > 0) status = "shortfall"
else if (excess > 0) status = "excess"
```

Compliance Result Structure

```
interface ComplianceResult {
  municipality: string
  municipalityId: string
  program: string
  required: number // Sites required based on population
  actual: number // Sites currently assigned
  shortfall: number // required - actual (if positive)
  excess: number // actual - required (if positive)
  complianceRate: number // (actual / required) * 100
  status: "compliant" | "shortfall" | "excess"
}
```

Site Counting Rules

Only the following sites count toward compliance:

- **Status:** "Active" or "Scheduled"
- **Program Match:** Site's programs array includes the target program
- **Municipality Match:** Site is assigned to the municipality

```
const municipalitySites = sites.filter(
  (site) =>
```

```
site.municipality_id === municipality.id &&  
site.programs.includes(program) &&  
site.status === "Active"  
)
```

3. Tool A: Direct Service Offset

A one-time per year reduction of collection site requirements applied across all communities.

Location in Code

- `components/tool-a-direct-service-offset.tsx`

Purpose

Allows Product Care to manually input a percentage reduction that automatically reduces the number of required collection sites across all communities equally.

Calculation Formula

```
New Required = ceil(Required × (1 – percentage / 100))  
Minimum: 1 site (cannot be reduced below 1)
```

Code Implementation

```
const calculateNewRequired = (required: number, percentage: number): number  
=> {  
  if (required === 0) return 0  
  const reduction = required * (percentage / 100)  
  const newRequired = Math.ceil(required - reduction)  
  return Math.max(1, newRequired) // Minimum 1 site required  
}
```

Detailed Examples

Example 1: 10% Global Reduction for Paint Program

Community	Population	Base Required	Reduction Calc	New Required
Toronto	2,800,000	29	$\text{ceil}(29 \times 0.9) = \text{ceil}(26.1)$	27 sites
Mississauga	720,000	15	$\text{ceil}(15 \times 0.9) = \text{ceil}(13.5)$	14 sites
Hamilton	570,000	14	$\text{ceil}(14 \times 0.9) = \text{ceil}(12.6)$	13 sites
Small Town	5,000	1	$\text{ceil}(1 \times 0.9) = \text{ceil}(0.9) = 1 \text{ (min)}$	1 site

Step-by-step for Toronto (10% reduction):

1. Base required for Paint: $13 + \text{ceil}((2,800,000 - 500,000) / 150,000) = 13 + 16 = \mathbf{29 \text{ sites}}$
2. Apply 10% reduction: $29 \times (10/100) = 2.9$ reduction
3. New required: $\text{ceil}(29 - 2.9) = \text{ceil}(26.1) = \mathbf{27 \text{ sites}}$

Example 2: 25% Reduction for Lighting Program

Community	Population	Base Required	Reduction Calc	New Required
Toronto	2,800,000	80	$\text{ceil}(80 \times 0.75) = \text{ceil}(60)$	60 sites
Ottawa	1,000,000	44	$\text{ceil}(44 \times 0.75) = \text{ceil}(33)$	33 sites
Small Village	2,000	1	$\text{ceil}(1 \times 0.75) = \text{ceil}(0.75) = 1 \text{ (min)}$	1 site

Step-by-step for Toronto (25% reduction):

1. Base required for Lighting: $34 + \text{ceil}((2,800,000 - 500,000) / 50,000) = 34 + 46 = \mathbf{80 \text{ sites}}$
2. Apply 25% reduction: $80 \times (25/100) = 20$ reduction

3. New required: $\text{ceil}(80 - 20) = \text{ceil}(60) = \mathbf{60 \text{ sites}}$

Example 3: Per-Community Override

Community	Population	Base	Global 10%	Override %	Final Required
Toronto	2,800,000	29	27	15% override	$\text{ceil}(29 \times 0.85) = \mathbf{25 \text{ sites}}$
Mississauga	720,000	15	14	No override	14 sites
Hamilton	570,000	14	13	5% override	$\text{ceil}(14 \times 0.95) = \mathbf{14 \text{ sites}}$

Features

- 1. **Global Percentage:** Apply same reduction to all communities
- 2. **Per-Community Override:** Edit individual community percentages
- 3. **Year & Program Selection:** Offsets are specific to year and program
- 4. **Persistence:** Saves to `direct_service_offsets` and `community_offsets` tables

Important Notes

- The offset is applied BEFORE compliance is calculated
- Minimum 1 site is always required (cannot reduce below 1)
- Offsets are program-specific (e.g., 10% for Paint doesn't affect Lighting)

4. Tool B: Event Application

Allows temporary events to offset site shortfalls in communities.

Location in Code

- `components/tool-b-event-application.tsx`

Purpose

Communities with shortfalls can use scheduled events to partially meet their site requirements.

Key Constraints

1. **Maximum Event Offset:** 35% of total required sites

```
const MAX_EVENT_OFFSET_PERCENTAGE = 35
const maxEventsAllowed = Math.floor(totalRequired *
(MAX_EVENT_OFFSET_PERCENTAGE / 100))
```

2. **Shortfall Calculation** (excludes events):

```
const communitySites = sites.filter(
  (s) =>
    s.municipality_id === community.id &&
    s.programs.includes(selectedProgram) &&
    (s.status === "Active" || s.status === "Scheduled") &&
    s.site_type !== "Event" // Events excluded from base count
)
const shortfall = Math.max(0, required - communitySites.length)
```

3. **Event Eligibility:**

- Site type must be "Event"
- Status must be "Active" or "Scheduled"
- Must match the selected program

Workflow

1. System identifies communities with:
 - Shortfall > 0 (need more sites)
 - Available events > 0 (have events to apply)
2. User can:

- **Edit Events:** Select which events to apply per community
- **Apply All:** Automatically apply events across all communities (respects 35% cap)

3. Applied events reduce the shortfall:

```
shortfall = Math.max(0, shortfall - appliedEvents.length)
```

Statistics Tracked

- **Events Remaining:** Total events not yet applied
- **Offsets Remaining:** Sum of remaining shortfalls across all communities
- **Max Allowed:** 35% of total required sites

Detailed Examples

Example 1: Calculating Maximum Events Allowed

Scenario: 5 communities with Paint program requirements

Community	Population	Required Sites
Toronto	2,800,000	29
Mississauga	720,000	15
Hamilton	570,000	14
Ottawa	1,000,000	17
London	420,000	11
Total		86 sites

Maximum Events Calculation:

```
Max Events = floor(86 × 0.35) = floor(30.1) = 30 events
```

Example 2: Applying Events to Communities with Shortfalls

Before Events:

Community	Required	Actual (non-event)	Shortfall	Available Events
Hamilton	14	10	4	3
London	11	8	3	2
Ottawa	17	15	2	4

After Applying Events:

Community	Required	Actual	Shortfall	Events Applied	New Shortfall
Hamilton	14	10	4	3	$\max(0, 4-3) = 1$
London	11	8	3	2	$\max(0, 3-2) = 1$
Ottawa	17	15	2	2 (of 4 available)	$\max(0, 2-2) = 0$

Total Events Applied: 7 (within 30 max allowed)

Example 3: Apply All Respecting 35% Cap

Scenario: Total required = 50 sites, Max events = $\text{floor}(50 \times 0.35) = 17$ events

Community	Shortfall	Available Events	Events to Apply	Remaining Capacity
Community A	5	6	$\min(5, 6, 17) = 5$	$17 - 5 = 12$
Community B	8	10	$\min(8, 10, 12) = 8$	$12 - 8 = 4$
Community C	6	5	$\min(6, 5, 4) = 4$	$4 - 4 = 0$
Community D	3	4	$\min(3, 4, 0) = 0$	STOPPED

Result: 17 events applied (maximum reached)

5. Tool C: Adjacent Community Reallocation

Allows excess sites from one community to count toward an adjacent community's requirements.

Location in Code

- `components/tool-c-adjacent-reallocation.tsx`
- `lib/reallocations.ts`

Purpose

Communities with excess sites can share them with neighboring communities that have shortfalls.

Eligibility Rules

Sites that CAN be reallocated:

- Site type: "Collection Site" (NOT "Event")
- Operator type: Retailer, Distributor, Private Depot, Product Care, Other

Sites that CANNOT be reallocated:

```
const NON_REALLOCATABLE_OPERATORS = [  
  "Municipal",  
  "First Nation/Indigenous",  
  "Regional District"  
]
```

Calculation Process

1. Calculate shortfalls for all communities:

```
const shortfallMap: Record<string, number> = {}
communities.forEach((community) => {
  const required = calculateRequirement(community.population, program)
  const actual = sites.filter(...).length
  const shortfall = required - actual
  if (shortfall > 0) shortfallMap[community.id] = shortfall
})
```

2. Find communities with eligible excess:

```
const excess = communitySites.length - required
const eligibleSites = communitySites.filter(
  (s) => s.site_type !== "Event" &&
    !NON_REALLOCATABLE_OPERATORS.includes(s.operator_type)
)
const eligibleExcess = Math.min(excess, eligibleSites.length)
```

3. Identify adjacent communities with shortfalls:

- Uses adjacency map (from `adjacent_communities` table)
- Only shows adjacent communities that have shortfalls

Reallocation Rules by Program

EEE (Lighting) Program:

- Sites can ONLY be reallocated to **directly adjacent** municipalities

HSP Programs (Paint, Solvents, Pesticides):

- Sites can be reallocated to:
 - Adjacent municipalities, OR
 - Municipalities within the same upper-tier region

```
// From lib/reallocations.ts
if (program === "Lighting") {
  // EEE: adjacent only
  if (!adjacentMunicipalities.includes(toMunicipality.name)) {
    errors.push("EEE (Lighting) sites can only be reallocated to adjacent municipalities")
  }
} else if (["Paint", "Solvents", "Pesticides"].includes(program)) {
```

```
// HSP: adjacent or same upper-tier
const isAdjacent = adjacentMunicipalities.includes(toMunicipality.name)
const isSameUpperTier = fromMunicipality.region === toMunicipality.region
if (!isAdjacent && !isSameUpperTier) {
  errors.push("HSP sites can only be reallocated to adjacent or same
upper-tier")
}
}
```

Percentage Limits

- **Event reallocations:** Maximum 35% of required sites
- **Site reallocations:** Maximum 10% of required sites

```
if (reallocation.reallocation_type === "event" && percentage > 35) {
  errors.push("Events can offset maximum 35% of required sites")
}
if (reallocation.reallocation_type === "site" && percentage > 10) {
  errors.push("Adjacent community sharing limited to 10% of required sites")
}
```

Detailed Examples

Example 1: Identifying Eligible Excess

Scenario: Vaughan has 8 Paint sites but only requires 5

Site Name	Operator Type	Site Type	Eligible?
Home Depot Vaughan	Retailer	Collection Site	✓ Yes
Rona Vaughan	Retailer	Collection Site	✓ Yes
City of Vaughan Depot	Municipal	Collection Site	✗ No (Municipal)
Spring Paint Event	Private Depot	Event	✗ No (Event type)
Benjamin Moore	Distributor	Collection Site	✓ Yes

Calculation:

- Total sites: 8
- Required: 5

- Excess: $8 - 5 = 3$
- Eligible sites: 3 (Home Depot, Rona, Benjamin Moore)
- Eligible excess: $\min(3, 3) = 3$ **sites can be reallocated**

Example 2: Reallocation to Adjacent Community

Scenario: Vaughan (excess) → Markham (shortfall)

Community	Required	Actual	Status	Adjacent?
Vaughan	5	8	Excess: 3	Source
Markham	6	4	Shortfall: 2	✓ Adjacent to Vaughan
Richmond Hill	4	3	Shortfall: 1	✓ Adjacent to Vaughan
Toronto	29	25	Shortfall: 4	✓ Adjacent to Vaughan

Reallocation Options for Vaughan's 3 eligible excess sites:

1. Reallocate 2 to Markham (fills shortfall completely)
2. Reallocate 1 to Richmond Hill (fills shortfall completely)
3. Cannot reallocate all 3 to Toronto (would exceed 10% limit: $\text{floor}(29 \times 0.10) = 2 \text{ max}$)

Example 3: Program-Specific Reallocation Rules

Lighting (EEE) - Adjacent Only:

From	To	Adjacent?	Same Region?	Allowed?
Vaughan	Markham	✓ Yes	Yes	✓ Allowed
Vaughan	Barrie	✗ No	No	✗ Not Allowed
Vaughan	Newmarket	✗ No	Yes (York)	✗ Not Allowed

Paint (HSP) - Adjacent OR Same Upper-Tier:

From	To	Adjacent?	Same Region?	Allowed?
Vaughan	Markham	✓ Yes	Yes (York)	✓ Allowed
Vaughan	Newmarket	✗ No	Yes (York)	✓ Allowed (same region)
Vaughan	Barrie	✗ No	No	✗ Not Allowed

6. Data Models

Municipality (Community)

```
interface Municipality {
  id: string
  name: string
  population: number
  tier: "Single" | "Lower" | "Upper"
  region: string
  province: string
  census_year?: number
}
```

Collection Site

```
interface CollectionSite {
  id: string
  name: string
  address: string
  municipality_id: string
  site_type: string // "Collection Site" | "Event"
  operator_type?: string // See OPERATOR_TYPES
  programs: string[] // ["Paint", "Lighting", etc.]
  status: "Active" | "Inactive" | "Scheduled" | "Pending" | "Deactivated"
  latitude?: number
  longitude?: number
}
```

Reallocation

```
interface Reallocation {
  id: string
  site_id: string
  from_municipality_id: string
  to_municipality_id: string
  program: string
  reallocation_type: "site" | "event" | "direct_return"
  percentage: number
  status: "pending" | "approved" | "rejected"
}
```

```
    rationale?: string
  }
```

Operator Types

```
const OPERATOR_TYPES = [
  "Retailer",
  "Distributor",
  "Municipal",
  "First Nation/Indigenous",
  "Private Depot",
  "Product Care",
  "Regional District",
  "Regional Service Commission",
  "Other"
]
```

7. Business Rules & Constraints

Site Counting Rules

Rule	Description
Status Filter	Only "Active" or "Scheduled" sites count
Program Match	Site must include the target program
Event Exclusion	Events are counted separately (Tool B)

Offset Limits

Tool	Maximum Offset
Tool A (Direct Service)	No hard limit (percentage-based)
Tool B (Events)	35% of required sites
Tool C (Adjacent Reallocation)	10% of required sites

Reallocation Eligibility

Operator Type	Can Be Reallocated?
Retailer	✔ Yes
Distributor	✔ Yes
Private Depot	✔ Yes
Product Care	✔ Yes
Other	✔ Yes
Municipal	✘ No
First Nation/Indigenous	✘ No
Regional District	✘ No

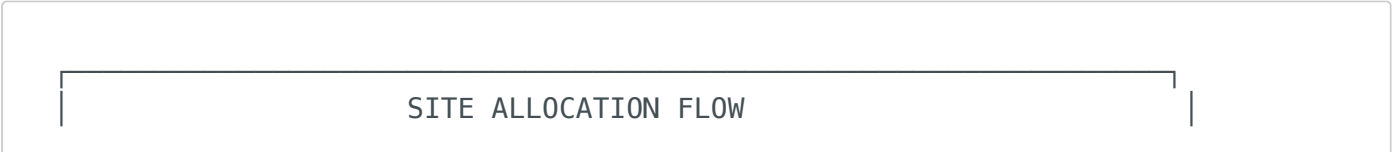
Program-Specific Rules

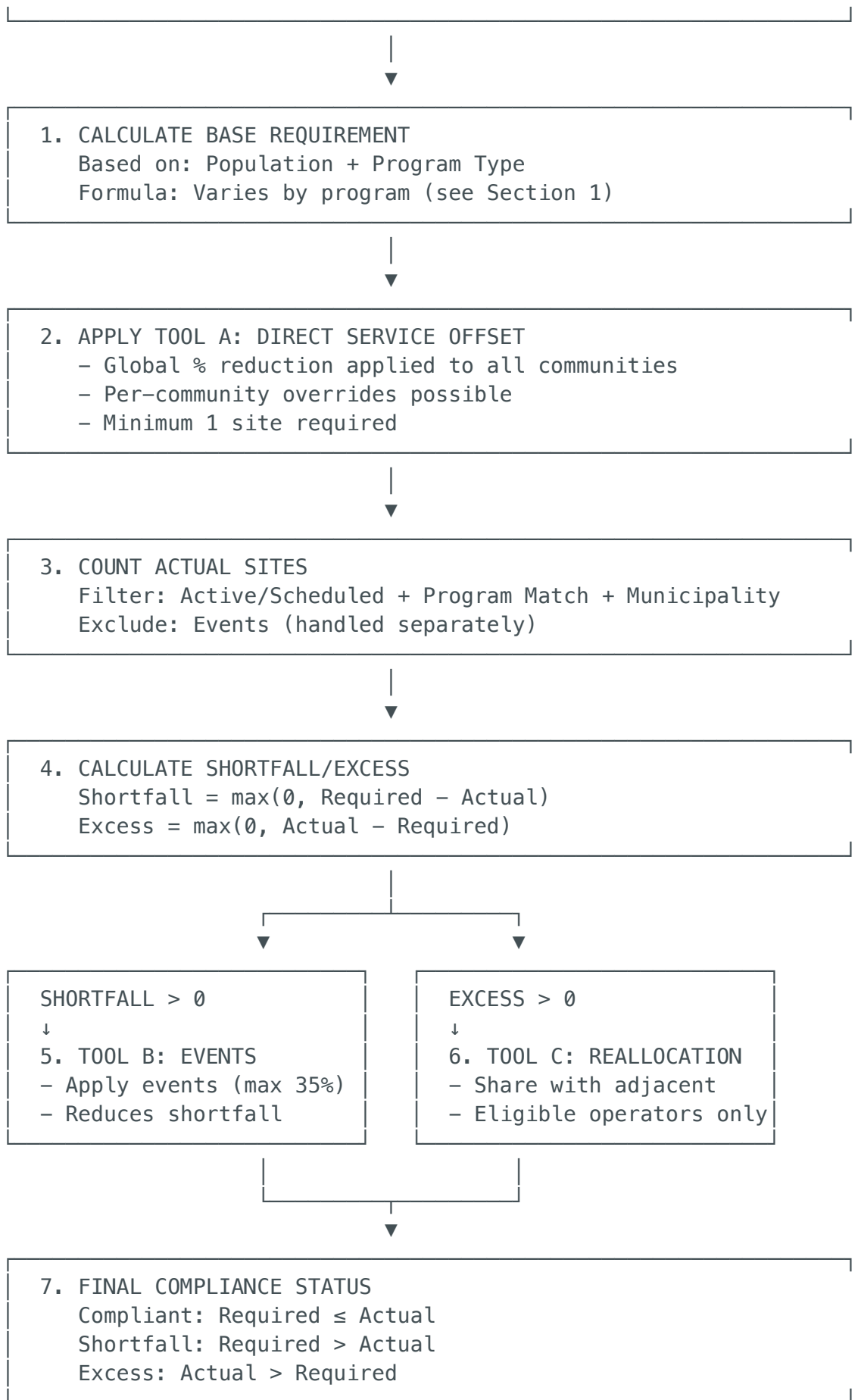
Program	Reallocation Scope
Lighting (EEE)	Adjacent communities only
Paint (HSP)	Adjacent OR same upper-tier region
Solvents (HSP)	Adjacent OR same upper-tier region
Pesticides (HSP)	Adjacent OR same upper-tier region

Minimum Requirements

- Communities requiring at least 1 site cannot be reduced below 1 site (Tool A)
- Shortfall and excess values cannot be negative (use `Math.max(0, value)`)

Summary Flowchart





8. Complete Calculation Examples

End-to-End Example: Paint Program Compliance for York Region

This example walks through the complete calculation process for the Paint program across multiple communities in York Region.

Step 1: Calculate Base Requirements

Community	Population	Formula	Calculation	Base Required
Vaughan	320,000	$\text{ceil}(\text{pop} / 40000)$	$\text{ceil}(320000 / 40000) = \text{ceil}(8)$	8 sites
Markham	350,000	$\text{ceil}(\text{pop} / 40000)$	$\text{ceil}(350000 / 40000) = \text{ceil}(8.75)$	9 sites
Richmond Hill	200,000	$\text{ceil}(\text{pop} / 40000)$	$\text{ceil}(200000 / 40000) = \text{ceil}(5)$	5 sites
Newmarket	90,000	$\text{ceil}(\text{pop} / 40000)$	$\text{ceil}(90000 / 40000) = \text{ceil}(2.25)$	3 sites
Aurora	60,000	$\text{ceil}(\text{pop} / 40000)$	$\text{ceil}(60000 / 40000) = \text{ceil}(1.5)$	2 sites
Total	1,020,000			27 sites

Step 2: Apply Tool A Direct Service Offset (10% Global)

Community	Base Required	Offset Calc	New Required
Vaughan	8	$\text{ceil}(8 \times 0.9) = \text{ceil}(7.2)$	8 sites
Markham	9	$\text{ceil}(9 \times 0.9) = \text{ceil}(8.1)$	9 sites
Richmond Hill	5	$\text{ceil}(5 \times 0.9) = \text{ceil}(4.5)$	5 sites

Community	Base Required	Offset Calc	New Required
Newmarket	3	$\text{ceil}(3 \times 0.9) = \text{ceil}(2.7)$	3 sites
Aurora	2	$\text{ceil}(2 \times 0.9) = \text{ceil}(1.8)$	2 sites
Total	27		27 sites (minimal reduction due to ceiling)

Note: With small numbers, 10% reduction often results in same requirement due to ceiling function

Step 3: Count Actual Sites (Excluding Events)

Community	Collection Sites	Status Filter	Actual Count
Vaughan	Home Depot, Rona, City Depot, Benjamin Moore, RONA+, Sherwin Williams, Canadian Tire, Lowes, Dulux	Active: 9	9 sites
Markham	Home Depot, Rona, City Depot, Canadian Tire, Lowes	Active: 5	5 sites
Richmond Hill	Home Depot, City Depot, Rona	Active: 3	3 sites
Newmarket	Canadian Tire, Home Depot	Active: 2	2 sites
Aurora	Home Depot	Active: 1	1 site

Step 4: Calculate Shortfall/Excess

Community	Required	Actual	Shortfall	Excess	Status
Vaughan	8	9	0	1	Excess
Markham	9	5	4	0	Shortfall
Richmond Hill	5	3	2	0	Shortfall
Newmarket	3	2	1	0	Shortfall

Community	Required	Actual	Shortfall	Excess	Status
Aurora	2	1	1	0	Shortfall
Total	27	20	8	1	

Step 5: Apply Tool B Events

Available Events:

Community	Events Available	Shortfall
Markham	2 (Spring Event, Fall Event)	4
Richmond Hill	1 (Community Day)	2
Newmarket	1 (Town Fair)	1
Aurora	0	1

Maximum Events Allowed: $\text{floor}(27 \times 0.35) = \text{floor}(9.45) = \mathbf{9 \text{ events}}$


Apply Events:


Community	Shortfall	Events Applied	New Shortfall
Markham	4	2	2
Richmond Hill	2	1	1
Newmarket	1	1	0 
Aurora	1	0 (none available)	1
Total	8	4	4

Step 6: Apply Tool C Adjacent Reallocation

Vaughan's Excess Analysis:



- Excess: 1 site
- Eligible sites: Check operator types

Site	Operator Type	Eligible?
City Depot	Municipal	 No

Site	Operator Type	Eligible?
Home Depot	Retailer	 Yes



Eligible excess: 1 site (Home Depot can be reallocated)

Adjacent Communities with Shortfalls:

- Markham: Shortfall 2, Adjacent 
- Richmond Hill: Shortfall 1, Adjacent 

Reallocation Decision: Reallocate Home Depot Vaughan → Markham

After Reallocation:

Community	Required	Actual	Shortfall	Status
Vaughan	8	8 (9-1 reallocated)	0	 Compliant
Markham	9	8 (5+2 events+1 reallocated)	1	Shortfall
Richmond Hill	5	4 (3+1 event)	1	Shortfall
Newmarket	3	3 (2+1 event)	0	 Compliant
Aurora	2	1	1	Shortfall

Final Compliance Summary

Metric	Value
Total Required	27 sites
Total Actual (after tools)	24 sites
Total Shortfall	3 sites
Communities Compliant	2/5 (40%)
Events Applied	4
Sites Reallocated	1

Quick Reference: Calculation Cheat Sheet

Base Requirement Formulas

```
PAINT:
  pop < 1000      → 0
  1000 ≤ pop < 5000 → 1
  5000 ≤ pop ≤ 500K → ceil(pop / 40,000)
  pop > 500K      → 13 + ceil((pop - 500K) / 150,000)

LIGHTING:
  pop < 1000      → 0
  1000 ≤ pop ≤ 500K → ceil(pop / 15,000)
  pop > 500K      → 34 + ceil((pop - 500K) / 50,000)

SOLVENTS/PESTICIDES:
  pop < 1000      → 0
  1000 ≤ pop < 10K  → 1
  10K ≤ pop ≤ 500K  → ceil(pop / 250,000)
  pop > 500K      → 2 + ceil((pop - 500K) / 300,000)
```

Key Limits

```
Tool A (Direct Service Offset):
- No hard cap on percentage
- Minimum 1 site always required
- Formula: ceil(required × (1 - %/100))

Tool B (Events):
- Maximum: 35% of total required sites
- Formula: floor(totalRequired × 0.35)

Tool C (Reallocation):
- Maximum: 10% of required sites per community
- EEE: Adjacent only
- HSP: Adjacent OR same upper-tier region
- Excluded operators: Municipal, First Nation/Indigenous, Regional District
```

File References

File	Purpose
lib/compliance.ts	Core compliance calculation functions

File	Purpose
lib/reallocations.ts	Reallocation validation and CRUD
lib/supabase.ts	Type definitions and constants
components/tool-a-direct-service-offset.tsx	Direct Service Offset UI & logic
components/tool-b-event-application.tsx	Event Application UI & logic
components/tool-c-adjacent-reallocation.tsx	Adjacent Reallocation UI & logic
components/compliance-analysis.tsx	Compliance dashboard & analysis
scripts/01-create-tables.sql	Database schema

Last Updated: December 2025